TESE DE DOUTORAMENTO

DEEP LEARNING FOR SMALL OBJECT DETECTION

Presentada por:

Brais Bosquet Mera

Dirixida por:

Manuel Mucientes Molina
Víctor M. Brea Sánchez

ESCOLA DE DOUTORAMENTO INTERNACIONAL

PROGRAMA DE DOUTORAMENTO EN INVESTIGACIÓN EN TECNOLOXÍAS DA INFORMACIÓN

Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS)

SANTIAGO DE COMPOSTELA
16 de outubro de 2020

DECLARACIÓN DO AUTOR DA TESE
DEEP LEARNING FOR SMALL OBJECT DETECTION

Don Brais Bosquet Mera

Presento a miña tese, seguindo o procedemento adecuado ao Regulamento, e declaro que:

1. *A tese abarca os resultados da elaboración do meu traballo.*
2. *No seu caso, na tese faise referencia ás colaboracións que tivo este traballo.*
3. *A tese é a versión definitiva presentada para a súa defensa e coincide coa versión enviada en formato electrónico.*
4. *Confirmo que a tese non incorre en ningún tipo de plaxio de outros autores nin de traballos presentados por min para a obtención de outros títulos.*

En Santiago de Compostela, 16 de outubro de 2020

Asdo. Brais Bosquet Mera

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

AUTORIZACIÓN DO DIRECTOR/TITOR DA TESE
DEEP LEARNING FOR SMALL OBJECT DETECTION

Dr. Manuel Mucientes Molina, Profesor Contratado Doutor da Área de Ciencia da Computación e Intelixencia Artificial da Universidade de Santiago de Compostela

Dr. Víctor M. Brea Sánchez, Profesor Titular da Área de Electrónica da Universidade de Santiago de Compostela

INFORMAN:

Que autorizamos a presentación desta tese, que se corresponde co traballo realizado por ***Don Brais Bosquet Mera*** *baixo a nosa dirección, considerando que reúne os requisitos esixidos no Regulamento de Estudos de Doutoramento da USC, e que como directores desta non incorre nas causas de abstención establecidas na Lei 40/2015.*

En Santiago de Compostela, 16 de outubro de 2020

Asdo. Manuel Mucientes Molina	Asdo. Víctor M. Brea Sánchez
Director da tese	Director da tese

A Ramón Bosquet,
de quien nunca habría tenido tiempo de aprender lo suficiente.

Those who have a 'why' to live, can bear with almost any 'how'.

Friedrich Nietzsche

Agradecementos

En primeiro lugar, quero expresar o meu máis sincero agradecemento aos meus directores Manuel Mucientes Molina e Víctor M. Brea Sánchez, pola confianza e os consellos que me brindaron durante estes anos.

Quixera agradecer tamén ao Departamento de Electrónica e Computación, así como ao Centro de Investigación en Tecnoloxías Intelixentes (CiTIUS), da Universidade de Santiago de Compostela, pola excelente contorna de traballo e os recursos que fixeron posible o desenvolvemento desta tese. Ao director adxunto do CiTIUS durante a maior parte da miña etapa, Prof. Paulo Félix, polo seu apoio e proximidade, á actual dirección do centro, Prof. Paula López e Prof. Senén Barro, así como ás unidades de apoio do centro pola súa constante dispoñibilidade. Tamén gustaríame mencionar ao Dr. Lorezo Seidenari, ao Prof. Alberto del Bimbo e a todo o grupo do MICC (*Media Integration and Communication Center*) da Universidade de Florencia por acollerme durante a miña estadía.

Non podía faltar un agradecemento especial as amizades que fixen durante esta etapa no CiTIUS, destacando a Mauro, como un irmán dende que estou en Santiago, pero tamén a Dani, Alberto, Lorenzo e Adrián, por estar sempre para todo o necesario.

Doutra banda, gustaríame agradecer a todas as entidades que fixeron posible o desenvolvemento desta tese co seu financiamento. Comezando polo programa de Axudas de apoio á etapa predoutoral da Xunta de Galicia (ref. ED481A-2016/078), e á Consellería de Educación, Universidades y Formación Profesional de la Xunta de Galicia (ref. ED431C 2018/29, ED431C 2017/69 e acreditación 2016-2019, ED431G/08, ambas co-financiadas polo Fondo Europeo de Desenvolvemento Rexional do programa ERDF/FEDER. Tamén expresar o meu agradecemento ao Ministerio de Economía, Industria y Competitividad (ref. TIN2017-84796-C2-1-R e TEC2015-66878-C3-3-R) e ao Ministerio de Ciencia, Innovación y Universidades (ref. RTI2018-097088-B-C32). Finalmente, quixera agradecer a colaboración de Gradiant e a cesión de GPUs por parte de NVIDIA.

16 de outubro de 2020

Resumo

A detección de obxectos é un dos principais campos de investigación da visión por computador. Ésta defínese como a identificación e encadre de todos os obxectos presentes nunha imaxe ou nun fotograma de vídeo. A detección de obxectos lévase a cabo en dúas etapas: (i) a localización do obxecto, onde se determina mediante unha caixa (*bounding box*) o máis axustada posible que determina a posición exacta do obxecto dentro da imaxe; e (ii) a clasificación do obxecto, onde ao obxecto asígnaselle unha categoría específica. A detección de obxectos é esencial para multitude de aplicacións, como a detección e recoñecementos faciais, a recuperación de imaxes, a seguridade, a vídeovixilancia, o control de tráfico, os vehículo autónomos, os procesos de verificación de identidade, a detección de anomalías en imaxes médicas ou a inspección de maquinaria e infraestruturas. Ademáis, a detección de obxectos irrompeu con forza nunha ampla gama de industrias, con casos de uso que van dende a seguridade persoal ata a produtividade no lugar de traballo. Hoxe en día seguen existindo importantes desafíos no campo do recoñecemento de obxectos, onde as posibilidades son infinitas cando se trata de aplicacións futuras que esixen un entendemento autónomo dos obxectos da contorna.

Nas aplicacións descritas existen obxectos de tamaños moi pequenos, case indistinguibles, pero cuxa identificación e detección é de gran importancia para as aplicacións. Algúns dos exemplos onde a importancia de identificar estes obxectos é destacable son: os sistemas dos vehículos automatizados ou as aplicacións como *sense and avoid* en vehículos aéreos non tripulados (UAVs, *Unmanned Aerial Vehicles*), que requiren a detección dun obxecto canto antes; a análise de imaxes por satélite, onde a práctica totalidade dos obxectos teñen un tamaño de só uns poucos píxeles; a inspección de infraestruturas onde, por razóns de seguridade, tense que informar da máis mínima imperfección; as imaxes médicas, nas que se ten que identificar a menor anomalía; a vixilancia de tráfico, que require detectar todos os obxectos,

independentemente da súa distancia; ou a vídeovixilancia, onde non se pode pasar por alto ningún obxecto. Con todo, é inevitable que canto máis pequenos sexan os obxectos, máis complexa sexa a súa detección. Isto pódese ver nos resultados obtidos sobre as bases de datos de detección de obxectos máis populares, como é MS COCO.

Durante varias décadas, o interese da comunidade científica na detección de obxectos estivo activo, pero non foi ata o no último decenio cando se acadaron os resultados máis destacables. Uns resultados obtidos, principalmente, grazas ao auxe das redes neuronais convolucionais (CNNs ou ConvNets, *Convolutional Neural Networks*) como extractores de características automaticamente adaptables, a mellora dos algoritmos de aprendizaxe profunda, e a transferencia da aprendizaxe a novos ámbitos. Os algoritmos tradicionais de detección de obxectos baseábanse principalmente en características deseñadas a man mediante diferentes algoritmos, e en clasificadores de aprendizaxe automática. Normalmente, estaban compostos de tres etapas: propostas de rexións prometedoras, extracción de características e clasificador. Os métodos de proposta de rexións prometedoras máis populares eran os baseados en xanelas deslizantes, computacionalmente ineficientes e pouco precisas para diferentes escalas. Máis tarde, desenvolvéronse métodos máis sofisticados para mellorar o rendemento, como a busca selectiva, que xera propostas de rexións fusionando píxeles próximos semellantes. A etapa de extracción de características estaba dominada por descriptores de características diferentes segundo o ámbito, como SIFT para puntos característicos, HOG para orientacións do gradiente ou Harris para a detección de esquinas. Por último, cada rexión clasifícase nunha categoría mediante un modelo de aprendizaxe supervisada como son as máquinas de soporte vectorial (SVM) ou os perceptróns multicapa (MLP). Adicionalmente, co fin de mellorar a precisión da detección, era común o uso de técnicas para procesar as características base, como a análise de compoñentes principais (PCA), métodos de agrupación (*clustering*) ou as bolsas de palabras (BoW, *Bag-of-Words*).

O punto de inflexión produciuse en 2012, cando Krizhevsky *et al.* definiron AlexNet, unha rede neuronal convolucional para a clasificación de imaxes que é capaz de aprender de forma automática as características das 1.000 categorías presentes no desafío ImageNet, mellorando, con moito, os algoritmos baseados nos enfoques tradicionais. É certo, con todo, que as técnicas de aprendizaxe profunda aplicadas ás imaxes baseadas nas redes neuronais convolucionais foron propostas a finais dos anos 80 por Yan LeCun (LeNet-5), amosando o seu potencial no recoñecemento de díxitos manuscritos (base de datos MNIST). Aínda que as arquitecturas AlexNet e LeNet-5 baseábanse nos mesmos operadores —convolucións, operadores de reducción (*pooling*) e funcións de activación—, o seu rexurdimento foi impulsado pola mellora computacional das GPUs que, xunto coa aparición de grandes bases de datos, fixeron posible o deseño e o adestramento de redes neuronais cada vez máis profundas. As técnicas de extracción de características baseadas en aprendizaxe profunda conduciron a notables avances na área de procesamento de imaxes. A aprendizaxe profunda mellora notablemente a extracción de características, facéndoa unha técnica clara e intuitiva, capaz de automatizar a adaptación a diferentes dominios cuxo deseño manual anteriormente requiría anos de investigación.

A primeira CNN que adaptou os avances na clasificación de imaxes á detección de obxectos foi Overfeat, lanzada un ano máis tarde que AlexNet, onde unha xanela deslizante a múltiples escalas envía rexións a unha CNN que as clasifica. Ademáis de demostrar que as CNNs eran eficientes para a detección de obxectos, tamén melloraron o acerto en clasificación de imaxes. Pouco máis tarde, apareceu R-CNN (*Regions with CNN features*) que apuntaría cara unha liña de investigación que prevalece actualmente. Os autores propuxeron un modelo que utiliza a busca selectiva como selector de rexións, a CNN como extractora de características, e un clasificador e un regresor que procesarían cada unha das rexións. Aínda que as CNNs como extractores de características ou *backbones*, en inglés, superaban considerablemente os métodos deseñados a man, seguían sendo computacionalmente ineficientes para a detección de obxectos, xa que o proceso debía repetirse para cada rexión da imaxe. Para abordar o problema apareceu Fast R-CNN, onde as rexións prometedoras son esta vez procesadas directamente sobre un mapa de características (*feature map*) profundo. Entón, a CNN como extractora de características é aplicada a toda a imaxe e unha nova capa denominada *RoI pooling*, en inglés, selecciona as zonas do mapa de características profundo asociadas a cada rexión, mapeando a saída en vectores do mesmo tamaño para que poidan ser introducidos en capas totalmente conectadas (*fully connected layers*) para a súa final clasificación e regresión. Neste punto, unicamente a etapa de proposta de rexións prometedoras non se leva a cabo mediante un enfoque de aprendizaxe profunda, minguando o rendemento do conxunto. A solución a esta cuestión dividiu, a partir dese momento, á comunidade científica en dúas estratexias principais: detectores baseados en rexións en dúas etapas (*two-stage region-based detectors*) e detectores que predín directamente as rexións a partir dos mapas de características (*one-shot detectors*).

- **Detectores baseados en rexións**: este enfoque baséase nunha rede de propostas de rexións prometedoras (RPN, *Region Proposal Network*) que utiliza mapas de características profundas adestrados para aprender a xerar posibles rexións de interese e envialas, nunha segunda etapa, á clasificación de obxectos e á regresión de rexións. Esta RPN foi proposta por Ren *et al.* en Faster R-CNN como unha evolución natural de Fast R-CNN.

- **Detectores nunha etapa**: este enfoque obtén as rexións prometedoras directamente dos mapas de características, en lugar de ter unha rede independente. Este deseño foi proposto en primeiro lugar por Redmon *et al.* coa rede neuronal YOLO. Este enfoque concreto divide a imaxe nunha cuadrícula e a cada unha das celas dalle unha probabilidade de existencia dun obxecto xunto coas correspondentes coordenadas da rexión. Outra interesante proposta é SSD, que actúa de forma similar a YOLO pero utiliza múltiples escalas por cada unha das celas para mellorar a predición.

Posteriormente, realizáronse multitude de melloras en cada un dos enfoques. Un dos avances máis significativos no campo da detección de obxectos é a rede FPN (*Feature Pyramid Network*). FPN é unha arquitectura deseñada para detectar obxectos de diferentes tamaños, aproveitando as diferentes escalas dos mapas de características. FPN componse de dúas direccións de execución: unha de abaixo cara arriba e outra de arriba cara abaixo. A execución de abaixo cara arriba é a dirección convencional para a extracción de características. A medida que subimos, a resolución espacial diminúe e as características identificadas son máis complexas, é dicir, o valor semántico de cada capa aumenta. O camiño de arriba a abaixo permite construír capas de maior resolución pero dotadas dunha maior semántica. A estes mapas de características de diferente resolución, engádeselles unha RPN a cada nivel permitindo identificar e localizar obxectos a diferentes escalas. A FPN está relacionada coa SSD no sentido de que a SSD tamén propón rexións a diferentes escalas de mapas de características pero, a diferenza da FPN, sen as características de alto nivel obtidas mediante a vía descendente. Sen elas, os mapas de características de menor profundidade na SSD conteñen só características de baixo nivel que non son eficaces para a detección de obxectos, prexudicando a detección de obxectos pequenos. A arquitectura FPN foi adoptada por outras solucións como RetinaNet para detectar obxectos nunha soa etapa, eliminado as RPNs e colocando dúas subredes, unha de clasificación e outra de regresión. Os detectores de CNN baseados en rexións como a rede FPN demostraron ser máis exitosos cando o tempo de cálculo non é esencial, destacando aínda máis cando se detectan obxectos pequenos.

Aínda que os actuais detectores de obxectos da CNN proporcionan unha gran precisión nunha ampla gama de escalas, a eficiencia da detección de obxectos pequenos é considerablemente inferior á detección de obxectos máis grandes, o que abre camiño a unha maior mellora. Os principais problemas da detección de obxectos pequenos son dous: (i) as arquitecturas actuais reducen a resolución a medida que aprenden características en capas de máis alto nivel, o cal é contraproducente cando o obxecto é pequeno, posto que as súas características poden perderse nese proceso, e (ii) as bases de datos máis populares como MS COCO ou ImageNet centran a súa atención en tamaños de obxecto superiores. Cabe destacar que nos últimos anos a detección de obxectos pequenos experimentou un progreso significativo seguindo o camiño que propuxo FPN, ao demostrar que é necesario utilizar ao mesmo tempo mapas de características de alto valor semántico e gran resolución. Moitos autores propoñen diferentes formas de combinar capas con diferentes resolucións e crear un único ou varios bloques convolucionais con información de características de alto e baixo nivel, como son *HyperNet*, SDP (*Scale-Dependent Pooling*) ou ION (*Inside-Outside Net*). Outro exemplo é MDFN, unha CNN nunha soa etapa que propón explotar unicamente as capas profundas pero introducindo módulos *Inception* con filtros a múltiples escalas para mellorar tanto a información semántica como a contextual. Neste traballo demóstrase como o contexto é moi relevante para a detección de obxectos pequenos. En referencia á escaseza de bases de datos con obxectos pequenos, ésta foise paliando nos últimos anos grazas a imaxes gravadas con UAVs sobre amplas áreas cunha considerable cantidade de obxectos pequenos, destacando UAVDT e VisDrone.

Actualmente, é moi común atoparse con bases de datos en vídeo ou aplicacións que requiren procesamento de vídeo. Os enfoques definidos anteriormente son capaces de localizar os obxectos fotograma a fotograma, pero non son capaces de aproveitar a coherencia temporal entre obxectos, útil para mellorar o rendemento final. Ademáis, os vídeos expoñen problemas adicionais que dan lugar a clasificacións inestables como, por exemplo, desenfoque de movemento, oclusións do obxecto ou cambios abruptos do punto de vista da cámara. Por conseguinte, traballos recentes dirixiron a súa atención á detección de obxectos en vídeo. Este aumento de interese produciuse, principalmente, dende a chegada do desafío ImageNet-VID en 2015, seguida por outras bases de datos en vídeo como UA-DETRAC, YouTubeObjects, ou os xa mencionados UAVDT e VisDrone.

As técnicas de aprendizaxe profunda que implican a información de varios fotogramas para mellorar a detección dun obxecto no momento actual dan lugar ás denominadas CNNs espazo-temporais. Un problema directamente relacionado coa detección de obxectos en vídeo, e do que se nutriu nos seus comezos, é o campo de recoñecemento de accións, onde as propostas baseadas en dous fluxos de entrada (*two-streams*) se converteron no enfoque estándar. Nestas solucións, ás características profundas usuais nas CNNs que traballan con imaxes engádeselles un novo fluxo de características que definen o movemento con respecto aos fotogramas veciños. Como resultado do seu éxito, este método adaptouse á detección de obxectos en vídeo para identificar correspondencias entre obxectos a través do tempo. Por exemplo, tanto na rede FGFA (*Flow-Guided Feature Aggregation*) como en MANet (*Motion-Aware Network*), as correspondencias temporais baséanse na información dun fluxo óptico. Outro enfoque interesante para incrementar a precisión final é a mellora das deteccións actuais mediante unha análise das puntuacións obtidas nos obxectos de fotogramas veciños. O método de asociación de obxectos durante varios fotogramas denomínase *object linking*, e o resultado da asociación de cada obxecto coñécese como *tubelet*. Neste caso, para a asociación de deteccións é común a utilización de solape, fluxo óptico ou operadores de correlación, e a obtención dos *tubelets* mediante algoritmos de *tracking* ou algoritmos para atopar as secuencias máis probables —por exemplo, o algoritmo de Viterbi.

Doutra banda, o adestramento dun modelo de aprendizaxe profunda significa, a grandes liñas, axustar os seus parámetros para que sexa capaz de asignar a unha entrada algunha saída, por exemplo, asignar a unha imaxe unha categoría. As CNNs de última xeración adoitan ter millóns de parámetros para poder recoñecer a ampla variabilidade que existente dentro dunha categoría. O problema é que, para adestrar tal cantidade de parámetros, requírese unha cantidade proporcional de exemplos. En xeral, canto maior sexa a variabilidade do adestramento, mellor será o rendemento final. Por conseguinte, nun escenario no que o número de instancias dos obxectos é reducido, é interesante considerar unha solución para aumentar o número de instancias sen ter que realizar a custosa tarefa de anotar os datos manualmente. Esta técnica coñécese como aumento de datos, do inglés, *data augmentation*. O seu obxectivo é incrementar o número e a diversidade dos datos de adestramento sen engadir novos datos como tal, senón engadindo exemplos existentes lixeiramente modificados ou creando exemplos sintéticos baseados en datos existentes. No caso da detección de obxectos, existen dous tipos principais: modificacións básicas de imaxes ou xeración de datos artificiais.

As modificacións básicas de imaxes son lixeiras alteracións dos datos existentes e, ao ser pouco custosas, son moi comúns á hora de adestrar un modelo de aprendizaxe profunda. Estas funcións básicas van dende cambios de escala, translación ou rotación das imaxes ata o recorte de segmentos da imaxe, a imaxe espello (*mirroring*) ou o modificacións na gama de cores.

A xeración de datos artificiais comprende enfoques máis sofisticados que teñen por obxecto aprender características relevantes dos datos de adestramento co fin de xerar novas instancias sintéticas. Un enfoque sinxelo é o de copiar e pegar instancias de obxectos en novos lugares para aumentar a variabilidade do contexto. Para que o obxecto sexa consistente coa contorna, AdaResampling, por exemplo, calcula un mapa de contexto, e logo recorta e reescala o obxecto de interese para colocalo de acordo co fondo. Con todo, demostrouse que as funcións de re-escalado introducen artefactos que están lonxe das características dos datos reais e, por tanto, son ineficientes para axudar a xeneralizar o modelo.

Os recentes avances na xeración de datos artificiais baséanse no principio da aprendizaxe antagónica. A aprendizaxe antagónica é unha metodoloxía que trata de adestrar un modelo para atopar posibles escenarios nos que un modelo adestrado erraría. Unha vez que se coñecen os casos en que fracasaría, é posible engadilos ao modelo para facelo máis robusto. Explorando estes principios, Goodfellow *et al.* marcaron un novo fito en 2014 coa formulación das populares redes xenerativas antagónicas (GANs, *Generative Adversarial Networks*). Estes enfoques constan de dous compoñentes: un xerador e un discriminador. Iterativamente, o xerador produce datos falsos e o discriminador trata de distinguir entre as imaxes reais e as falsas, e unha función de custo antagónica empuxa ao xerador para producir exemplos cada vez máis próximos aos reais. Esta idea foi aplicada para o ámbito da clasificación de imaxes mediante as denominadas DCGAN (*Deep Convolutional Generative Adversarial Network*).

Dende ese momento houbo unha gran cantidade de traballos que demostraron, e seguen demostrando, o potencial das GAN en multitude de ámbitos. Por poñer algúns exemplos: xeráronse datos artificiais para a base de datos MNIST que permitiron aumentar a precisión do modelo; DAGAN (*Data Augmentation GAN*) é capaz de aprender unha gran familia de transformacións para producir exemplos artificiais usando un codificador-decodificador (*encoder-decoder*); DeLiGAN logra xerar imaxes para varias modalidades diferentes en escenarios de reducido número de datos de adestramento; ou as xa famosas CycleGAN, compostas por dúas GAN cunha función de custo que outorga consistencia de ciclo e permite transferir o estilo dunha conxunto de datos a outro —por exemplo, o cambio de verán a inverno ou as condicións meteorolóxicas nun vídeo.

Tendo en conta este contexto, o obxectivo principal desta tese é abordar os inconvenientes da detección de obxectos pequenos mediante a definición de modelos capaces de mellorar o rendemento non só os obxectos pequenos —inferiores a 32 $\times$ 32 píxeles, como define MS COCO—, senón tamén os obxectos extremadamente pequenos. Cualitativamente, referímonos a obxectos extremadamente pequenos como aqueles que, se non fóra polo contexto circundante, apenas seríamos capaces de asignalos a unha categoría. Cuantitativamente, extremadamente pequeno refírese a tamaños inferiores a 16×16 píxeles. En particular, preténdense deseñar arquitecturas baseadas en aprendizaxe profunda centradas en localizar obxectos pequenos, e de definir sistemas para abordar o escaso número de instancias obxectos pequenos nas bases de datos máis populares. Para acadalo, propuxéronse os seguintes obxectivos específicos:

1. **Detección de obxectos pequenos en imaxes**. Deseño dunha arquitectura de aprendizaxe profunda para detección de obxectos capaz de traballar con mapas de características de alta resolución e alto valor semántico ao mesmo tempo sen aumentar o custo computacional. Ademáis, componse unha nova base de datos centrada en obxectos extremadamente pequenos para complementar a validación dos modelos propostos sen nesgo de tamaño.

2. **Detección de obxectos pequenos en vídeos**. Definición dunha arquitectura espazo-temporal baseada no deseño anterior. A nova arquitectura debe aproveitar a coherencia temporal dos vídeos para obxectos pequenos e mellora con ela as decisións de detección final.

3. **Xeración de datos artificiais para a detección de obxectos pequenos**. Deseño dun algoritmo de xeración automática de imaxes artificiais para incrementar o número de instancias de obxectos pequenos nunha base de datos dada. O sistema ten que ser capaz de xerar obxectos cuxas características axuden ao proceso de adestramento, mellorando a precisión do modelo obtido. Ademáis, as novas instancias deben dispoñerse de forma coherente na imaxe.

Nesta memoria detállanse as contribucións do traballo desenvolvido e os resultados máis relevantes da experimentación.

En concreto, o Capítulo 2 presenta STDnet, unha arquitectura baseada en redes neuronais convolucionais para a detección de obxectos pequenos, que é capaz de traballar con mapas de características de alta resolución e gran semántica nas capas máis profundas da rede, sen aumentar o custo computacional. STDnet basea o seu éxito nunha nova rede de rexións con contexto (RCN, *Region Context Network*), un módulo situado nas primeiras etapas da rede que está adestrado para seleccionar as zonas máis prometedoras da imaxe, é dicir, aquelas que conteñen idealmente un obxecto xunto co seu contexto. A este módulo engádeselle unha capa de agrupación de rexións (RCL, *RoI Collection Layer*) que encapsula estas rexións nun único mapa de características. A partir deste punto, a rede unicamente procesa as zonas seleccionadas e, por tanto, os novos mapas de características poden manter unha alta resolución e maior velocidade durante toda a execución sen sobrecargar a memoria. Ademáis, faise unha proposta básica sobre un enfoque espazo-temporal (STDnet-bST) que se estenderá no Capítulo 3. Finalmente, xérase unha base de datos de detección de obxectos pequenos (USC-GRAD-STDdb), unha nova base de datos de vídeo para a detección de pequenos obxectos para validar os modelos propostos. USC-GRAD-STDdb contén 115 segmentos de vídeo en alta resolución con máis de 56.000 obxectos anotados de tamaños comprendidos entre 4×4 e 16×16 píxeles.

O Capítulo 3 describe STDnet-ST, unha CNN espazo-temporal baseada en STDnet e STDnet-bST para a detección de obxectos pequenos de vídeo. STDnet-ST presenta dúas etapas: unha primeira etapa que correlaciona as rexións máis prometedoras de dous fotogramas consecutivos para buscar coincidencias, e un método de asociación de rexións que compón *tubelets* de alta calidade. Concretamente, STDnet-ST consiste en dúas ramas de STDnet unidas por un operador de correlación que traballa coas rexións propostas polas RCNs. Por iso, a pesar de que os obxectos pequenos teñen poucas características distintivas, o contexto dentro das rexións fai posible que o operador de correlación poida recoñecer os obxectos en dous instantes temporais. A probabilidade de correlación denota o nivel de similitude por par de rexións e permite facer coincidir os obxectos aproveitando a coherencia temporal do vídeo. STDnet-ST presenta tamén un método de asociación de rexións baseado no algoritmo de Viterbi que utiliza as probabilidades de correlación para vincular as deteccións de ambos os fotogramas. Por último, a nosa proposta inclúe un algoritmo de supresión de *tubelets* (*tubelet suppression*) que detecta as asociacións menos rendibles xerando novos nodos e conseguindo, con iso, unicamente *tubelets* de gran calidade ao detectar aqueles que probablemente non teñen boas deteccións. A toma de decisión final utiliza a variabilidade de probabilidades dentro de cada *tubelet*.

No Capítulo 4 detállase un sistema para a xeración artificial de obxectos pequenos en vídeo (*data augmentation*) que poboa, de forma automática, unha base de datos cun número maior de instancias de obxectos sintéticos pequenos de alta calidade. O compoñente principal é unha nova arquitectura baseada nas GANs (DS-GAN) adestrada para producir novos obxectos pequenos sintéticos utilizando a información de obxectos máis grandes. O xerador ten unha arquitectura de codificador-decodificador (*encoder-decoder*), onde a dimensión de entrada é maior que a de saída, co fin de xerar un pequeno obxecto sintético a partir dun obxecto real máis grande xunto cun vector de ruído. Estes obxectos sintéticos pequenos achéganse cada vez máis aos obxectos pequenos reais, xa que o xerador trata de enganar a un discriminador adestrado para diferenciar os pequenos obxectos sintéticos dos reais mediante a introdución de artefactos provenientes destes obxectos. Un vez xerados os obxectos pequenos artificiais, o sistema combina arquitecturas de segmentación de obxectos, fluxo óptico, *inpainting* e *blending* para integralos en posicións coherentes dentro dos fotogramas do vídeo. Concretamente, as técnicas de segmentación de obxectos empréganse para eliminar o contexto que rodea ao obxecto artificial. O fluxo óptico localíza posicións coherentes dentro da imaxe. As técnicas de *inpainting* empréganse para eliminar calquera obxecto dentro da posición seleccionada. Finalmente, as técnicas de *blending* melloran a adecuación espacial e de cor para que o obxecto se axuste ao novo fondo da forma máis natural posible.

A tese remata co Capítulo 5, onde se resumen as conclusións da investigación realizada, ademáis de discutir posibles liñas de traballo futuro que poderían permitir mellorar os resultados acadados.

Contents

Chapter 1

Introduction

1.1 Motivation

Object detection is one of the main topics in computer vision. It refers to correctly identifying and labeling all objects present in an image or video frame, and it consists of two steps: (i) object localization, in which a bounding box or enclosing region is determined as tight as possible to locate the exact position of the object in the image; and (ii) object classification, in which the located object is labeled to a specific category. Object detection is required for several practical applications like face detection and facial recognition, image retrieval, security, surveillance, traffic monitoring, self-driving cars, identity verification, medical imaging or machine and infrastructure inspection. Besides, object detection is breaking into a wide range of industries, with use cases ranging from personal security to productivity in the workplace. Today, significant challenges remain in the field of object recognition and the possibilities are endless when it comes to future applications that demand an automatic understanding of the objects in an environment.

Within each of the applications outlined above, there are instances of objects that are displayed in small, almost indistinguishable sizes, but whose understanding and detection is of great importance in several applications. To mention some specific examples: automated vehicle systems or applications such as sense and avoid in unmanned aerial vehicles (UAVs) need to detect an object as far as possible; satellite image analysis, where almost all objects are only a few pixels in size; infrastructure inspection, where for safety reasons the slightest imperfection has to be detected; medical imaging, where the slightest anomaly has to be identified; traffic monitoring, which requires to detect all the objects, no matter how far they

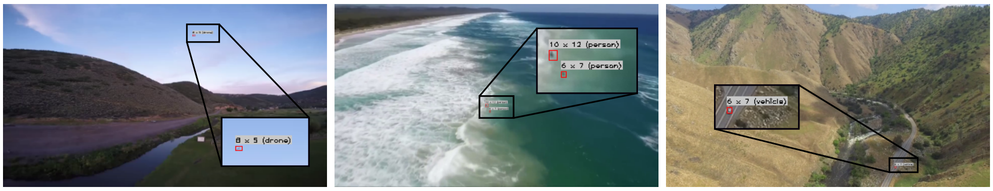


Figure 1.1: Annotated small objects in real-world images.

are; or surveillance, where objects of interest cannot be missed. Inevitably, however, the smaller the objects the lower the accuracy of the detection. This can be seen in popular object detection contests like MS COCO [90]. Figure 1.1 illustrates the challenge of small object detection in three different scenarios from the dataset created in this PhD Thesis. This is addressed in this PhD Thesis through three axes: (i) small object detection in static images; (ii) small object detection in video; and (iii) data augmentation for small object detection.

1.1.1 Image Object Detection

The scientific community's interest in image object detection has been active for several decades, but in the last one important results have been achieved, mainly thanks to the rise of deep convolutional neural networks (CNNs or ConvNets) as adaptive feature extractors and transfer learning as a method of transmission of prior knowledge. Traditional object detection algorithms mainly relied on handcrafted features, reached through feature engineering, and machine learning classifiers [33, 95, 126]. They based their operation on three different stages: region proposals, feature extraction, and region classifier (Figure 1.2). Region proposal methods were traditionally based on sliding windows [17, 21], which were computationally inefficient and less precise in a variety of scales. Later on, more sophisticated methods were developed to improve performance, such as selective search [125], which generates region proposals by merging similar pixels into regions. The feature extraction step was dominated by different handcrafted feature detectors such as SIFT for interesting points, HOG for gradient orientations or Harris for corner detection [21, 89]. Finally, each region was classified into a category by a supervised learning model like a support vector machine (SVM) or a multilayer perceptron (MLP) [65]. In order to improve the detection accuracy, different techniques have been used to process the features, like Principal Component Analysis (PCA), k-means or Bag-of-Words (BoW) [22].

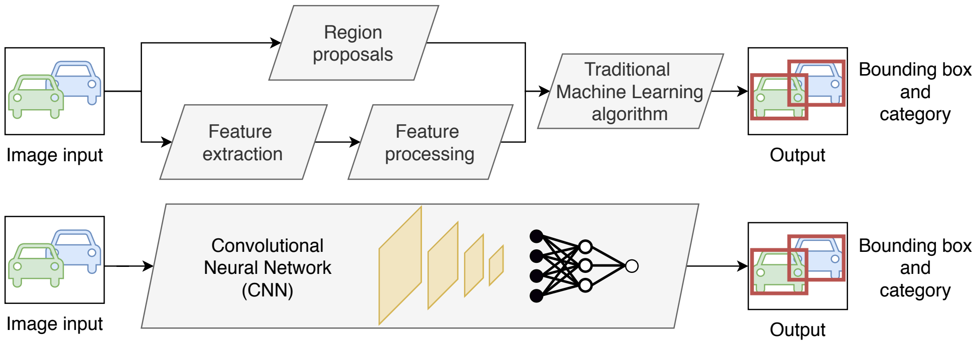

Region proposals
Image input
Feature extraction
Feature processing
Traditional Machine Learning algorithm
Output
Bounding box and category
Image input
Convolutional Neural Network (CNN)
Output
Bounding box and category

Figure 1.2: Block diagram of the object detection workflow: traditional object detectors (top) and CNN-based object detectors (bottom).

The breakthrough point was achieved in 2012 when Krizhevsky *et al.* defined AlexNet [72], a convolutional neural network for image classification capable of learning automatically the features of the 1,000 categories present in the ImageNet benchmark [110] and improving, by far, the algorithms based on traditional approaches. It is true, however, that deep learning techniques applied to images, based on convolutional neural networks, were proposed in the late 80s by Yan LeCun [73] where they proved their potential in the recognition of handwritten digits (MNIST dataset). Although the AlexNet and LeNet-5 [74] architectures were based on the same operators —convolutions, pooling and activation functions—, the resurgence of neural networks was also driven by the computational optimization of GPUs and the emergence of big data and dataset repositories, which made possible the design and training of deeper neural networks [30, 80, 110]. The deep learning-based techniques for feature extraction led to remarkable advances in the area of image processing. Automatic feature learning is a clear and intuitive technique able to automatize the adaptation to different domains whose manual design, previously required years of research.

The first deep neural network for object detection was Overfeat [112], released one year later than AlexNet, where a multi-scale sliding window approach feeds a CNN that showed the obvious, that the CNNs were efficient also for object detection, and that they improved image classification too. They were soon followed by R-CNN: regions with CNN features [40] that pointed out a line of research that is currently prevalent. The authors proposed a model that uses selective search [125] and each region was fed into a CNN, which produced a high dimensional feature vector. This vector was then used for the final classification and bounding box regression. Although CNNs as feature extractors or backbones considerably outperformed handcrafted methods, they were still computationally inefficient for object detection as the feature extraction step had to be repeated for each region of the image. To solve this issue, Fast R-CNN [39] generates the region proposals with selective search, just as R-CNN, but those regions are processed directly on a deep feature map applied to the entire input image. A novel Region of Interest (RoI) pooling selects the features related to each region and creates pooled same-size feature vectors that are fed into a fully connected network for classification and regression. At this point, only the region proposal stage was not conducted through a deep learning approach, reducing computational performance. The solution to incorporate the regions to the network divided, from this moment on, the computer vision community in two strategies: region-based detectors (*two-shot* or *two-stage*) and detectors that directly predict boxes from feature maps (*one-shot* or *one-stage*).

- **Region-based detectors**: this approach is based on a Region Proposal Network (RPN) to generate regions of interest using deep feature maps in the first stage and send the region proposals down the pipeline for object classification and bounding-box regression in the second stage. This RPN was proposed by Ren *et al.* [106] with Faster R-CNN as a natural evolution of Fast R-CNN.

- **One-shot detectors**: this approach treats "*object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities*" [102], instead of a classification problem relying on region proposals. This design was firstly proposed by Redmon *et al.* [102] with the YOLO network that divides the image into a grid, where each cell predicted a confidence score of an object being present with the corresponding bounding box coordinates. Another network that predicts classes and bounding boxes at once is the Single Shot Detector (SSD) [83], which is comparable to YOLO but uses multiple aspect ratios per grid cell and different feature map scales to improve prediction.

Subsequently, a large number of innovations have been made in each of the above approaches [37, 46, 103]. One that brought a significant advance in the object detection field was the Feature Pyramid Network (FPN) [78]. FPN is a backbone designed for detecting objects in different scales by taking advantage of the different feature map scales, or pyramid of feature maps. FPN composes of a bottom-up and a top-down pathway. The bottom-up pathway is the usual convolutional network for feature extraction. As we go up, the spatial resolution decreases. With more high-level structures detected, the semantic value for each layer increases. The top-down pathway permits to construct higher resolution layers from a semantic rich layer. With those semantic rich feature maps at different scales, an RPN is added to each pyramid level to capture objects at different scales. FPN is related to SSD in that SSD also proposes regions at different feature map scales but, unlike FPN, without the high-level features gathered by the top-down pathway. Without them, the shallower feature maps in SSD contain only low-level features that are not effective for accurate object detection, thus harming small object detection. The same FPN-based architecture was adopted by RetinaNet [79] that removes the RPNs and adds two subnetworks —class subnet and bounding box subnet— to detect objects in one-stage. The main improvement is obtained through a novel loss function (Focal Loss) to address the class imbalance in one-shot detectors. Region-based CNN detectors have proven to be more successful when computation time is not essential, being even more prominent when referring to small objects.

As stated before, current CNN object detectors provide high accuracy at a wide range of scales. Nevertheless, small object detection accuracy lags behind that of larger objects [82], which opens the way for more improvement. The problems of detecting such small objects are twofold: (i) deep CNNs architectures commonly reduce the input resolution to understand high-level features, which is counterproductive when the object is so small that it may be lost along the way, and (ii) the most popular image datasets such as MS COCO [80] or ImageNet [72] focus their attention on larger objects.

Even though, over the last years, the scope of small object detection has witnessed significant progress since FPN demonstrated that it is required to exploit semantic rich and coarse feature maps at the same time to properly detect small objects. In [139], authors propose a scale-dependent pooling along with layer-wise cascade rejection classifiers in several branches for the different object sizes. Then, if an object proposal has a height lower than 64 pixels, the regions are processed by a scale-dependent pooling with more resolution than if they were larger objects. References [4], [71] or [45] combine layers with different resolutions to create a single convolutional block that gathers the information from upper layers into the bottom ones. Another approach is MDFN [85], a one-shot CNN that proposes only to exploit high-layers and, at the same time, improves the small and occluded object detection. This is done by introducing inception modules with multi-scale filters to enhance both the semantic and contextual information. Here it is shown that context is quite relevant for detecting small objects.

The objective of this PhD Thesis is to address the drawbacks in small object detection by designing a novel deep learning architecture focused on small object detection able to work with high resolution feature maps and high semantic value at the same time, without increasing the computational cost. Particularly, the architecture must be able to detect not only small objects —under 32×32 as defined in MS COCO—, but also extremely small objects. Qualitatively, we refer to extremely small as those objects without definitive visual cues to assign them to a category, if it were not for the surrounding context (Figure 1.1). Quantitatively, extremely small refers to sizes under 16×16 pixels. The lack of specific datasets with small objects has been partially addressed with the rise of UAVs with built-in cameras to record wide areas in the wild with small objects and good quality such as UAVDT [27] and VisDrone2019-VID [153]. Still, as the amount of small objects in public datasets is reduced, it will be mandatory to generate a new dataset to validate the proposed model without size bias.

1.1.2 Video Object Detection

Video object detection aims to detect objects of different categories in video sequences. Basically, video object detection is a conventional image object detection where the images to be processed are each one of the video frames. However, video frames present some singular characteristics. On the one hand, temporal coherence is a useful cue to improve performance. On the other hand, they bring additional challenges that lead to unstable classifications for the same object across the video, e.g., motion blur, video defocus, long time partially or fully occlusion or severe camera view changes. Therefore, some research initiatives have been directed towards video object detection through the exploitation of temporal features.

Video object detection has also had a recent upturn, to the detriment of the popular static image benchmarks that dominated the object detection field (ImageNet [110] or MS COCO [80]). The advent of ImageNet video object detection challenge (ImageNet-VID) [110] followed by other video datasets like UA-DETRAC [132], YouTubeObjects [61] or those already mentioned UAVDT [27] and VisDrone2019-VID [153], set a new course in object detection.

Deep learning-based techniques that involve the information of several frames to improve the detection of an object at the current moment gives rise to the so-called spatio-temporal CNNs [14]. A related problem to video object detection is the action recognition field, in which the two-stream networks have become the standard approach [41, 97, 115]. In these solutions, the standard features computed by a CNN are augmented with another stream of features propagated from neighboring frames. As a result of its success, this feature propagation method has also been adapted to video object detection for finding correspondences between objects across time. In Flow-Guided Feature Aggregation (FGFA) [154], feature correspondences are found based on optical flow information. Wang *et al.* [129] propose Motion-Aware Network (MANet), which also bases the search of correspondences on optical flow. Alternatively, in [135], authors introduce a Spatial-Temporal Memory Module (STMM) that uses deep feature maps from several frames and the previous memory state to adjust the current spatio-temporal memory. Bertasius *et al.* [6] define a Spatio-Temporal Sampling Network (STSN) based on deformable convolutions to establish feature correspondences across the video.

There is another interesting approach that aims to improve the overall accuracy by modifying the detections and their scores in the current frame according to detection's scores in neighboring frames. The method of associating objects during several frames is commonly called object linking, and the result of the association of each object is known as a tubelet. The approach in [63] performs video object detection in the current frame and tracks the objects through neighboring frames based on the mean optical flow vector within boxes. Similarly, the approach in [62] links objects into long tubelets using a tracking algorithm and then adopts a classifier to aggregate the detection scores in the tubelets. The solution addressed in [32] inserts a correlation operator between two input frames to extract motion information of the objects across time. The correlation operates over the entire feature maps at different scales and estimates local feature similarity for various offsets between the two frames. Then, they link the detected objects into tubelets and re-weight the detections' scores within them. Correlating whole feature maps implies that, as an object becomes smaller, their movement represents a considerably smaller influence, even though the correlation acts on several scales. Another alternative for video object detection introduced in [121] proposes a modified RPN called Cuboid Proposal Network (CPN) for detecting objects in multiple input frames. The cuboid proposals are regressed and classified to create short tubelets. Consecutive short tubelets are merged into long tubelets by a linking algorithm that takes the best detection for each overlapping frame between two tubelets. Finally, Cores *et al.* [19] describe a temporal pooling operator that summarizes RoI pooled features linked through object tubelets to feed a spatio-temporal double head that takes advantage of both spatial and spatio-temporal information.

Given the direction towards the computer vision field is heading to, with both the release of video datasets and applications that increasingly demand video processing such as face detection, surveillance, self-driving cars or UAVs, the architecture designed during this PhD Thesis must be able to take advantage, when possible, of the temporal information to improve the final accuracy.

1.1.3 Data Augmentation

Training a deep learning model really means to tune its parameters so it is able to map an input into some output —e.g., an image into a category. State-of-the-art CNNs typically have millions of parameters in order to be able to recognise the wide variability that can exist within a category —e.g., type included into each category, translation, viewpoint, size or illumination. The problem is that, for training such a number of parameters, it is required a proportional amount of examples. In general, the greater the training variability the better the final performance, as the model becomes more generalized.

Therefore, in a scenario where the number of instances of the objects is reduced, it is crucial to consider a solution to increase the number of instances without having to perform the costly task of manually annotating data. This technique is known as data augmentation. The goal of data augmentation is to increase the number and diversity of the training data without adding new data, but by adding existing slightly modified examples or by creating synthetic examples based on existing data. For object detection, there are two main types of data augmentation: basic image manipulations and generative synthetic approaches.

Basic image manipulations are slight alterations to the existing data, so most of the deep learning designs usually combine many of them. For example, scale, translation, and rotation [18, 111, 127] or cropping, image mirroring and color processing [72] are very common for object recognition. For object detection, image mirroring and object-centric cropping are the most widely used [83, 116].

Generative synthetic approaches comprise more sophisticated models that aim to learn features from the training data to generate new synthetic instances. A straightforward approach is the copy-paste of object instances in new places to increase context variability. This is done in [69] by cropping small objects and randomly pasting them in the image. The weak point of this approach is that the object may not be consistent with the background. To deal with this, Chen *et al.* [15] design AdaResampling, a method that first understands the background by computing a context map, and then crops and re-scales the target object to be placed in accordance with the background. The issue here is more difficult to grasp, but [113] and

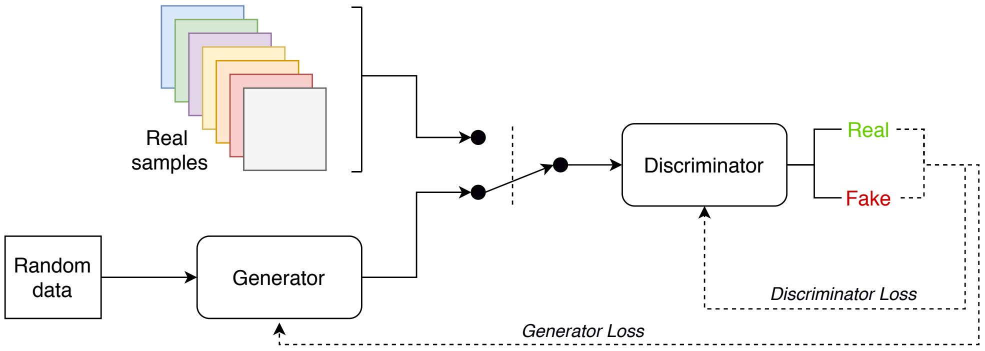


Figure 1.3: Generative Adversarial Network (GAN) overview. The generator attempts to create examples close to those in the real dataset to fool the discriminator. The discriminator tries to distinguish between real and generated examples.

[10] have proven that re-scaling functions introduce artefacts that are far from real-world data features and are, therefore, inefficient in helping to generalize the model.

Recent advances in data augmentation are based on the principle of adversarial learning. Adversarial learning is a methodology that tries to find possible scenarios where a trained model would fail through the training of another model with contrasting objectives. Once the instances where it would fail are known, it is possible to add them to the model to increase its robustness. In [130], authors train an adversarial network in parallel with Fast-RCNN to generate examples that would lead the detector to misdetection, so that the model will adapt itself to learn to classify these adversarial examples. Exploring the adversarial learning principles, Goodfellow *et al.* [42] first formulated the popular generative adversarial networks (GANs). GANs comprise two networks: the generator and the discriminator. Iteratively, the generator produces fake data and the discriminator tries to distinguish between real and fake images, and an adversarial loss pushes the generator to produce examples more and more closer to real ones (Figure 1.3). That idea was applied for image classification tasks in Deep Convolutional GANs (DCGAN) by Radford *et al.* [101].

Mirza and Osindero [86] extended the definition with a conditional input to generate synthetic images on MNIST dataset [75] along with their label. DAGAN (Data Augmentation GAN) [1] learns a large family of transformations to produce synthetic examples using a encoder-decoder as generator. The generator takes an input image, adds noise when it is encoded and, finally, decodes it. Reference [44] introduces DeLiGAN, a GAN-based framework that manages to generate images for a number of different modalities in low data scenarios. The key idea is that, instead of sampling directly from a simple latent distribution, authors reparameterize it using a mixture of Gaussian model. Finally, another interesting idea for data augmentation with GANs is CycleGAN [152], where authors propose two GANs with cycle consistency loss for style transfer from one setting to another –e.g., change from summer to winter or the weather conditions in a video.

Considering all the aforementioned work, data augmentation approaches become even more important when dealing with instances of objects where the number of training examples is reduced. Small object detection, as has been mentioned, falls short of a high number of instances within the most popular datasets, so data augmentation techniques are an interesting field to study. For this reason, in addition to releasing a small object dataset, this PhD Thesis presents an approach based on generative adversarial networks to augment the number small objects in existing popular datasets.

1.2 Objectives

The main goal of this PhD Thesis is to explore deep learning techniques with the objective of achieving state-of-the-art results on small object detection. Particularly, the aim is to design novel CNN architectures focused on finding small objects as well as to define systems to overcome the scarce number of small objects in public datasets. To achieve this, the following objectives have been pursued:

O1. A convolutional neural network for small object detection in images

The proposed convolutional neural network (CNN) architecture must be able to process the image information with a low resolution loss in order to keep most of the small objects features and thus identify the small objects in the image. A naive increase in the feature map dimension would dramatically increase computational costs, making the proposal infeasible. The hypothesis is that focusing only on some regions of the image would allow to keep high resolution feature maps in deep layers. This would overcome state-of-the-art architectures, which have a high stride, harming them for the detection of small objects. Furthermore, a new video database with small objects will be created to validate the proposals. In order to generalize the results, the database must contain various categories in different real-world environments.

O2. A spatio-temporal architecture for small object detection in video

The spatio-temporal CNN has to exploit the temporal coherence of the objects in videos to improve the object detection performance. This requires processing several frames simultaneously and using the information from the intermediate feature maps to associate the final detections and create temporal object tubelets. Once a new level of temporal abstraction is achieved for each object, it is possible to enhance the object detection in future frames based on previous positions and scores and, thus, to improve the overall performance.

O3. A data augmentation approach for generating synthetic small objects

The data augmentation approach will increase the number of instances of small objects in a given dataset. Therefore, the system must be able to automatically generate objects that do not exist in the dataset and place them coherently into the image —e.g., a ground object cannot appear in the sky. Moreover, the features of the generated synthetic small objects must help the training process, improving the obtained model precision.

1.3 Contributions

The research developed in this thesis has led to the following contributions:

C1. The design of Small Target Detection network (STDnet), a deep learning architecture for small object detection able to work with high resolution feature maps in the deepest layers without increasing the computational cost.

C1.1 STDnet relies its success on the novel Region Context Network (RCN), a module trained to select regions centered on an object together with its context, and the RoI Collection Layer (RCL) for integrating those disjoint regions. So that, the RCN points out the most promising regions of the image at early stages, and the RCL merges them into a single feature map. Then, from this point onwards, the network only has to process the selected regions and, therefore, the new feature maps can keep a high resolution with a lower memory overhead and a higher frame rate.

C1.2 The release of the Small Target Detection database (USC-GRAD-STDdb), a new video database for small object detection for validating STDnet and future proposals. It comprises 115 video segments and more than 56,000 annotated objects with sizes between 4×4 and 16×16 pixels.

C2. STDnet-ST, A STDnet spatio-temporal enhancement for video object detection that correlates the most promising regions of two consecutive frames and creates high quality object tubelets.

C2.1 STDnet-ST consists of two STDnet branches bounded together by a correlation operator working with the RCN. The correlation is performed over the most promising regions of both images resulting in a correlation score per pair. Although small objects have few distinguishing features, the context within the RCN regions makes possible for the correlation operator to be able to recognize objects in two temporal instants. The correlation score denotes the level of similarity per pair and allows to match objects by exploiting the video temporal coherence.

C2.2 The high quality object tubelets along the video are achieved by a new correlation-based tubelet linking. First, it uses the correlation scores to link the detections throughout the frames and uses the confidence variability of the tubelet to make the final decision. Then, a tubelet suppression algorithm avoids unprofitable tubelets by adding dummy nodes to the Viterbi algorithm based on the information coming from promising RCN regions without detections.

C3. A full pipeline for small object data augmentation in video that automatically populates an input dataset with high quality synthetic small objects.

C3.1 The main component is a novel Down-Sampling Generative Adversarial Network (DS-GAN) architecture trained to produce new synthetic small objects using the information of larger objects and incorporating artefacts coming from real-world small objects. The generator has a encoder-decoder architecture where the input dimension is larger than the output to generate a synthetic small object starting from a larger real object along with a noise vector. These synthetic small objects are increasingly close to real small objects as the generator tries to fool a discriminator trained to differentiate synthetic small objects from real ones.

C3.2 The pipeline combines and adapts state-of-the-art approaches in object segmentation, object inpainting and object blending to integrate the synthetic small objects generated by the DS-GAN in coherent positions within the video frames. Object segmentation techniques are used to remove the context around the objects generated by the DS-GAN so that they can be inserted properly into the new frame. The selected position within the image must first be cleaned by object inpainting if there is a real object overlapped. Finally, object blending is performed to enhance the spatial and color consistencies to make the object fit the new background and look as natural as possible.

Publications

All the contributions detailed above are included in the following publications, which comprise the scientific output of the research developed during the development of this thesis:

Journals

- **B. Bosquet**, M. Mucientes, and V. M. Brea. STDnet: Exploiting high resolution feature maps for small object detection. Engineering Applications of Artificial Intelligence, Vol. 91, No. 103615, 2020.
  Impact factor: 4.201 (JCR 2019).
  Journal ranked Q1 in JCR 2019.
  Category: COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE (33/136).
  Thesis contribution: C1.

- **B. Bosquet**, M. Mucientes, and V. M. Brea. STDnet-ST: Spatio-Temporal ConvNet for Small Object Detection. Article submitted to Pattern Recognition in March 2020 and currently under second round review.
  Impact factor: 7.196 (JCR 2019).
  Journal ranked Q1 in JCR 2019.
  Category: COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE (12/136).
  Thesis contribution: C2.

- **B. Bosquet**, L. Seidenari, V. M. Brea, M. Mucientes, and A. Del Bimbo. Data augmentation for small object detection through a downsampling GAN. Article submitted to IEEE Transactions on Image Processing in October 2020 and currently under review.
  Impact factor: 9.340 (JCR 2019).
  Journal ranked Q1 in JCR 2019.

Category: COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE (8/136).
Thesis contribution: C3.

- M. Fernández-Sanjurjo, **B. Bosquet**, M. Mucientes, and V. M. Brea. Real-time visual detection and tracking system for traffic monitoring. Engineering Applications of Artificial Intelligence, Vol. 85, pp. 410-420, 2019.
  Impact factor: 4.201 (JCR 2019).
  Journal ranked Q1 in JCR 2019.
  Category: COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE (33/136).

Conferences

- **B. Bosquet**, M. Mucientes, and V. M. Brea. STDnet: A ConvNet for Small Target Detection. *In Proceedings of the 29th British Machine Vision Conference (BMVC)*, p. 253, Newcastle upon Tyne (UK), 2018.
  Conference Ranking (GGS Rating): A, Class 2.
  Thesis contribution: C1.

- **B. Bosquet**, M. Mucientes, and V. M. Brea. Correlation-based ConvNet for Small Object Detection in Videos. *In Proceedings of the 25th International Conference on Pattern Recognition (ICPR)*, Milan (Italy), 2021.
  Conference Ranking (GGS Rating): A-, Class 2.
  Thesis contribution: C2.

Others

- EPO Patent: **B. Bosquet**, M. Mucientes, V. Brea, A computer-implemented method and system for detecting small objects on an image using convolutional neural networks. (Under review)

1.4 Dissertation structure

This dissertation is divided into five chapters. Chapters 2, 3 and 4 contain a detailed description of the contributions and experimental results, and Chapter 5 present the conclusions of this research. More concretely, the document structure is as follows:

- First, Chapter 2 introduces STDnet, a novel CNN architecture for small object detection, along with a public video dataset with small targets in real-life scenarios, overcoming the drawbacks of current datasets. STDnet includes a novel Region Context Network (RCN) that selects the most promising regions of the image and discards the remainder, and an RoI Collection Layer (RCL) that encapsulates these regions in a unique and reduced feature map. Thanks to these two techniques, the algorithm is able to remove overhead by not processing a large part of the image from that point onwards, allowing to work with high resolution and high semantic feature maps. Additionally, a spatio-temporal baseline network (STDnet-bST) is proposed, which will be further extended in Chapter 3. Finally, USC-GRAD-STDdb, a video dataset with more than 56,000 extremely small objects in complex scenarios in the wild, is released.

- Chapter 3 enhances STDnet and STDnet-bST by describing STDnet-ST, a novel spatio-temporal convolutional neural network that exploits temporal information for video small object detection. STDnet-ST simultaneously generates the detections of the current frame, together with the correlations between the current and previous frames. The correlation is performed in a natural way over the most promising regions of the image. Finally, STDnet-ST presents a novel tubelet linking able to increase the confidence of the detections most likely to be true positives within high quality tubelets, and decrease the confidence of those most likely to be false positives within unprofitable tubelets, thereby, improving the overall performance.

- Chapter 4 details a full pipeline for small object data augmentation. The pipeline takes a video dataset as input and returns the same dataset but with new synthetic objects. This is done by converting the visual features of real larger objects —which can be found in many datasets in a large number— into high quality synthetic small objects and, also, by placing them in adequate positions in an existing image. To do so, the pipeline comprises three main steps: (i) generation of small objects from large ones through DS-GAN, a generative adversarial network; (ii) search of an adequate position within the image through optical flow; (iii) small object integration via inpainting and blending techniques.

- Finally, in Chapter 5 the conclusions of the research developed in this thesis are summarized. Moreover, this chapter discusses future lines of work that might allow to improve the results.

CHAPTER 2

STDNET: EXPLOITING HIGH RESOLUTION FEATURE MAPS FOR SMALL OBJECT DETECTION

In this chapter we introduce our convolutional neural network approach for small object detection (STDnet). The motivation behind it comes from the fact that accuracy of small object detection with convolutional neural networks (ConvNets) lags behind that of larger objects. This is in part caused by the lack of specific architectures and datasets with a sufficiently large number of small objects. The work in this chapter aims at these two issues.

First, this chapter introduces STDnet for small object detection that we defined as those under 16×16 pixels. One of the main challenges of the design is the requirement to achieve high-resolution maps in deep layers so that objects, no matter how small, will be represented strongly throughout the execution. This is built on a novel early visual attention mechanism, called Region Context Network (RCN), to choose the most promising regions, while discarding the rest of the input image. Processing only specific areas allows STDnet to keep high resolution feature maps in deeper layers, providing low memory overhead and higher frame rates. Second, we also present USC-GRAD-STDdb, a video dataset with more than 56,000 annotated small objects in challenging scenarios.

The contents of this chapter are extracted from the following publications:

B. Bosquet, M. Mucientes, and V. M. Brea. STDnet: A ConvNet for Small Target Detection. *In Proceedings of the 29th British Machine Vision Conference (BMVC)*, p. 253, Newcastle upon Tyne (UK), 2018.

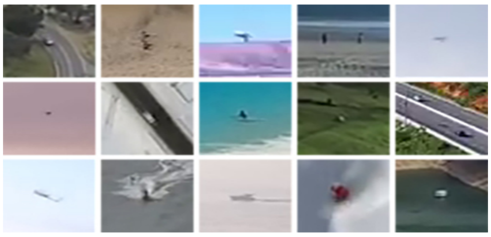
B. Bosquet, M. Mucientes, and V. M. Brea. STDnet: Exploiting high resolution feature maps for small object detection. Engineering Applications of Artificial Intelligence, Vol. 91, No. 103615, 2020.

2.1 Introduction

In the last years, solutions to visual object detection have experienced a fast evolution. This evolution goes from primary approaches based on machine learning [33, 95, 126] to deep learning techniques [47, 83, 103, 106], boosted by publicly available datasets with millions of annotated images [30, 80, 110].

Generic image and/or video datasets have become valuable benchmarks to assess the quality and advance of object detectors. Nevertheless, datasets for more specific scenarios or applications are also a need. This situation might lead to new state-of-the-art object detectors, sometimes cutting across different topics. As an example, a timely trend is to apply ideas from the visual object tracking field [34, 94] to exploit temporal features in video through spatio-temporal networks [14, 98].

In this line, applications like sense and avoid on board of unmanned aerial vehicles (UAVs) or video surveillance over wide areas demand early detections of objects to act quickly. This means to detect as far —and therefore small— an object as possible. Recent convolutional neural networks (ConvNets) object detectors, like the work in [78], provide high accuracy over a wide range of scales, from less than 32×32 pixels up to the image size. Qualitatively, we refer to small as those objects without definitive visual cues to assign them to a category. Quantitatively, small refers to sizes under 16×16 pixels. Figure 2.1a shows examples of this kind of objects where it can be seen that had it not been by the context around the foreground object, it would not be possible for a person to sort them out in a given category. Despite the existence of solutions focused on small objects, the most remarkable ones have been validated in face detection datasets [2, 52, 147, 148], but none of them have been tested in generic datasets such as MS COCO small objects subset ($<32\times32$ area) [80].


(a) Small objects as defined in this chapter, i.e., under 16 × 16 pixels.

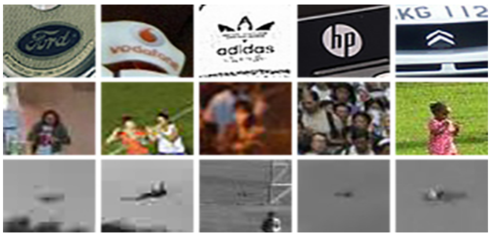

(b) Smallest objects available retrieved from public datasets.

Figure 2.1: Examples of 48 × 48 image patches from different databases with small objects.

When it refers to the scope of small targets, as those below 16 × 16 pixels as defined in this chapter, the available set of datasets presents several drawbacks as shown in Figure 2.1b. The top row shows small objects extracted from the FlickrLogos dataset presented in [60]. As apparent, this is a very specific image dataset. Also, their size exceeds that of 16 × 16 pixels, and it is possible to assign them to a given category. The middle row displays small objects from the image dataset introduced in [142]. Again, this dataset is very specific and the small objects like faces are actually part of a whole person —which although in the case of faces are of interest by themselves, this is not the general case. Finally, the last row shows samples from a video dataset addressed in [109]. These are monochrome and low quality videos, which, although relevant, are not the trend with the advent of ever better quality color image sensors. Another dataset with very specific small objects is [64].

As [82] point out, detecting small objects stands out as one of the key challenges in object detection. The main obstacle presented by state-of-the-art ConvNets is found at the region proposal level, which is inefficient when proposing valid regions for such small objects. This is given by the fact that the proposal generation of regions is applied on a very low resolution feature map, where the characteristics of small objects get too small to be detectable. One straightforward solution could be to modify a state-of-the-art ConvNet, disabling some backbone's downsampling to keep larger feature map resolutions. This, however, leads to large memory requirements, easily beyond the capacity of current high performance GPUs, and possibly to slower solutions. Another approach is to apply a region proposal generator directly in early stages of the network. However, this leads to the problem that the semantic information does not suffice to locate and classify the small objects appropriately.

This chapter introduces STDnet, a novel ConvNet architecture for small object detection, along with a public video dataset with small targets in real-life scenarios, overcoming the drawbacks of current datasets aforementioned above. STDnet takes advantage of the following hypothesis: being such small objects almost visually unrecognizable, they have simple characteristics that can be learned by the neural network at early stages, where the semantic information is low but enough to select zones of interest containing small objects. To implement this hypothesis, an early high resolution feature map feeds a novel promising regions extractor called Region Context Network (RCN), which does not have to delimit the objects perfectly, but larger regions —promising areas— which contain them. This set of disordered regions is encapsulated in a unique and reduced feature map by RoI Collection Layer (RCL). Thanks to these two techniques, the algorithm is able to remove overhead by not processing a large part of the image from that point onwards, allowing to work with high resolution feature maps. Finally, a common region proposal method takes the feature map of disordered regions as input to obtain the final bounding boxes. The STDnet approach allows to consume less memory than its counterparts for higher resolution of the last feature map.

The main contributions of our proposal are:

1. STDnet, a new ConvNet for small object detection able to work with high resolution feature maps in deepest layers. STDnet relies on two novel components, RCN and RCL, which work together to select the most promising areas of the image, generating a new single filtered feature map with them. Therefore, the filtered feature maps can keep a high resolution but with a lower memory overhead and a higher frame rate.

2. As STDnet has a final region proposal method that works with anchor boxes, we propose to automatically select the number and sizes of these anchors through a novel algorithm based on k-means. Our proposal differs from [103] in that our algorithm selects not only the sizes of the anchors, but also their number, making the anchors selection fully automatic.

3. A new video database, USC-GRAD-STDdb, for small object detection to cover the indicated dataset gap. USC-GRAD-STDdb presents more than 56,000 annotated objects of sizes between 4×4 and 16×16 (e.g., Figure 2.2). USC-GRAD-STDdb comprises 115 video segments (>25,000 frames) over the three principal landscapes: air, sea and land.

4. A spatio-temporal evolution/version of STDnet, called STDnet-bST, to exploit video information of USC-GRAD-STDdb.

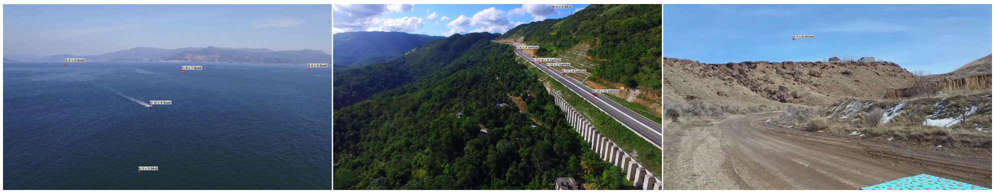


Figure 2.2: USC-GRAD-STDdb examples. Ground truth objects are enclosed in red boxes (best seen in color).

5. To bring STDnet closer to real on-board systems we have mapped both STDnet and STDnet-bST onto the embedded GPU Jetson TX2 through a series of computational optimizations.

2.2 Related Work

Modern object detectors are based on ConvNets [43, 54]. One-shot and two-stage solutions are the two main configurations of ConvNets adopted today. The former has two principal baselines, namely, SSD [83] and YOLO [103], which feature an excellent performance in computational cost and accuracy trade-off. As a drawback, both of them are outperformed by the two-stage approach when it refers to small objects [54, 82]. There are three main reasons for this: (1) the generation of the bounding boxes takes place in deep layers with low spatial resolution, which misses the opportunity of locating small objects —e.g., SSD and YOLO apply a downsampling of $8\times$ and $32\times$, respectively, to the original image to locate the smallest objects; (2) the use of a fixed sampling grid that works worse when objects are too close to each other or are too small and (3) the lower accuracy, at all scales, mostly produced by the great class imbalance between foreground objects and background proposals as these detectors evaluate $\approx 10^4$ candidate regions per image [79]. It is worth noting that the last drawback can be partially addressed by solutions such as hard negative mining [83] or max-out background label [148], and that RetinaNet [79] outperforms these solutions with a novel cost function. All the above and our own experiments have made our research focus on the two-stage solution.

The two-stage approach was popularized by R-CNN [40]. Its extension Faster-R-CNN [106] has become a milestone with the introduction of the Region Proposal Network (RPN). This approach, also known as region based object detectors, uses the RPN to generate a set of candidate object locations based on anchor boxes, which are predefined regions of different sizes and aspect ratios to cope with multiple scales. At a second stage, the backbone's upper layers are applied to classify the candidate locations into object of interest or background, besides refining the bounding box.

Sharing the same problem as one-shot detectors, the off-the-shelf Faster-R-CNN is not adequate for small object detection due to the fact that the global effective stride (GES) —downscaling of the input image with respect to the feature map that is the input to the region proposal method— is 16, which means that a 16×16 object is represented by just one pixel in that feature map. In addition, the anchor boxes are predefined manually and they were not conceived to handle such small objects. To tackle small objects, a finer GES is required. This leads to a very large memory overhead, making the implementation impossible for current GPUs[1].

In [20], authors propose an additional functionality to Faster-RCNN called Region-based Fully Convolutional Network (R-FCN). R-FCN exploits the different parts of a given object using position-sensitive maps. R-FCN generates $k \times k \times (C+1)$ feature maps —$k \times k$ object parts for C object categories plus background— instead of only one in the second stage detection. Additionally, R-FCN is fully convolutional, so it avoids the fully connected layers overhead. Still, this interesting improvement does not fix the GES problem and cannot be used to detect small objects, as their objects parts are very small, and as such indistinguishable in the input images.

The capability of dealing with objects of different sizes, and specially small ones, in Faster-R-CNN and R-FCN is limited due to low resolution feature maps and the few scales produced with the anchors as we outlined above. Hence, more recent ConvNets for object detection tackle scale invariance and small object detection with more elaborated solutions.

[139] propose a scale-dependent pooling along with layer-wise cascade rejection classifiers in several branches for the different object sizes. Then, if an object proposal has a height lower than 64 pixels, the regions are processed by a scale-dependent pooling with more resolution than if they were larger objects. Still, this proposal does not meet our needs for objects under 16×16 pixels.

In [77], authors focus on learning to transfer the information of small objects to similar large objects, what they call super-resolved objects. For this purpose they introduce a Perceptual Generative Adversarial Network for small object detection. The performance of this

[1]For reference, we use the NVIDIA Tesla P40 which has 24GB of memory.

approach has been tested on the Tsinghua-Tencent 100k dataset [155], considering small objects those smaller than 32×32, and on the Caltech benchmark [24], with pedestrians over 50 pixels tall, so it does not suffice for objects under 16×16 pixels.

[29] propose a Faster-R-CNN based approach for small objects. Authors validate the proposal in the FlickrLogos dataset [107] similar to the examples presented in Figure 2.1b (first row). The solution presents three levels of RPN which make use of feature maps with different resolution. To match different levels, high-level feature maps are upscaled through bilinear interpolation and then summed with the lower-level maps. Finally, classification and bounding box regression receive as inputs the combination of them. Similarly, in [12] several RPNs are proposed with the shallowest RPN working with the smallest objects. In the experimental evaluation the smallest object size ranges from 25 to 50 pixels of height, which does not suffice our needs.

An effective solution to detect objects in different scales, approached by [4], [71] or [45], is to combine layers with different resolutions throughout the ConvNet by creating a single convolutional block that gathers the information from upper layers into the bottom ones. A similar solution is adopted in [147] for face detection. This way, highly semantic information in wide receptive fields provided by the upper layers is combined with the low semantic information in narrow receptive fields. This combination is usually generated by skip connections, which discard some layers to connect more distanced ones. A single region proposal method takes this convolutional block as input to detect different scale objects.

Regarding the specific field of face detection, many advances have been made with outstanding results. In this line, [148] propose a modified SSD architecture which anticipates the object proposal to GES = 4, among other improvements. [2] employ a generative adversarial network (GAN) to improve the resolution of blurry small faces. Finally, [52] study the crucial influence of context in detecting small objects while proposing a model that uses specific scale templates to detect faces of different sizes —including very small ones.

Joining information from several convolutional layers and using several RPNs at different scales, [78] introduce Feature Pyramid Network (FPN). FPN builds a neural network that joins several levels of convolutional blocks through lateral connections. In each level junction, an RPN adapted naturally to a different object scale is applied. The FPN architecture features several RPNs at different GESs. The shallower convolution block —which is fed by the shallower combined feature map, with a GES = 4— is the RPN that detects the small objects, obtaining outstanding results. Since its release, the proposal has become a milestone, being currently adopted as baseline for the researches that lead the top entries of the MS COCO challenge [90].

Applying the improvements provided by FPN, RetinaNet [79] sheds light on how to improve performance metrics in the single-shot approach, including small objects. The work in [79] implements a simple FPN-based architecture that removes the RPNs and adds two subnetworks —class subnet and bbox subnet— to detect objects in one-stage. The main improvement is obtained through a novel loss function (Focal Loss) to address the class imbalance problem in single-shot detectors, leading to very promising results. The accuracy that reaches the proposal is similar to those obtained by the two-stage FPN, even in MS COCO small object scale ($<32\times32$ area).

This chapter focuses on small targets, i.e., under 16×16 pixels. Such small object targets make us different from the previous approaches: our sizes are significantly smaller than those of the above solutions and the categories we are dealing with are large objects –car, person, boat, etc.— but they are located at such a great distance that most of them do not feature definitive visual cues to classify them into a category (Figure 2.1a), making the object detection more difficult. Based on the knowledge from the aforementioned research, our STDnet architecture generates candidate object locations proceeding over the deepest layers, which exploits their semantic information, but keeping high feature map resolution —GES = 4— which allows to deal with small targets successfully with a reasonable memory overhead.

2.3 STDnet architecture

STDnet is an unified neural network approach proposed to detect small objects under 16×16 pixels. To address this, STDnet invests time in selecting promising zones at shallower layers to discard the rest of the input image and, thus, improving the overall computing performance. This allows to keep high resolution feature maps in these layers, especially favorable to detect small objects.

Figure 2.3 shows an overview of STDnet. It comprises five stages: early convolutions, Region Context Network (RCN), late convolutions, Region Proposal Network (RPN), and the classifier. Thereby, STDnet is presented as a Faster-R-CNN-based approach which employs ResNet-50 as a backbone —a good trade-off between accuracy, speed and GPU memory [47]. ResNet is represented as early convolutions that comprise the shallowest layers and late covolutions that encompass the deepest ones. The backbone can be any of the most widely

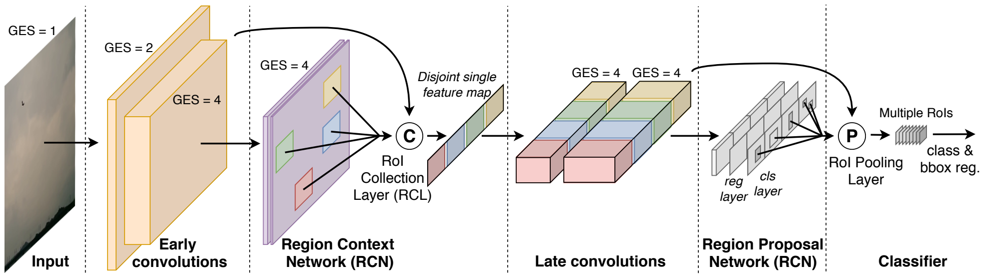


Figure 2.3: STDnet architecture. The RCN is placed after early convolutions to select only promising areas from the input image. Those zones are concatenated by the RCL so that the late convolutions act on the new feature map in a conventional way.

state-of-the-art solutions found in the literature —e.g., ResNet [47], DenseNet [53], VGG [116], etc.

Like most state-of-the-art ConvNets methods, STDnet begins to learn simple features from the objects of interest in shallower convolutional layers, named here as early convolutions. Unlike other methods, just after the shallower convolutions, STDnet applies a novel detector of promising areas over the shallower feature map, RCN, to select regions that most likely contain small objects. Then, top scored regions are gathered in a single feature map by RCL for the deeper convolutional layers —late convolutions— to act as usual on the disjoint areas. There is a caveat, so that the convolutions between the different areas do not affect each other, RCL introduces a null 1px size padding —since ResNet applies convolution filters no greater than 3 $\times$ 3. This padding must be reset to 0-padding after each convolution higher than 1 $\times$ 1. In the late convolutions stage, the memory saved by ruling out not promising areas can be used to keep the feature map in high resolution at the same time that the semantic information is increasing. STDnet applies a single RPN that takes as input the fourth convolutional block (C_4), which contains the most promising areas provided by the RCN but with richer semantic information. The RPN proposes as output the locations of the objects more precisely, performing bounding box regression and classification as object and background inside those RCN promising areas. These regions are further refined in a final classifier. This architecture differs from cascaded region proposal approaches [137, 150] in that the entire image information is not used during the whole computation, rather a new synthetic feature map is constructed only with the areas of the image that the RCN considers to be the most important.

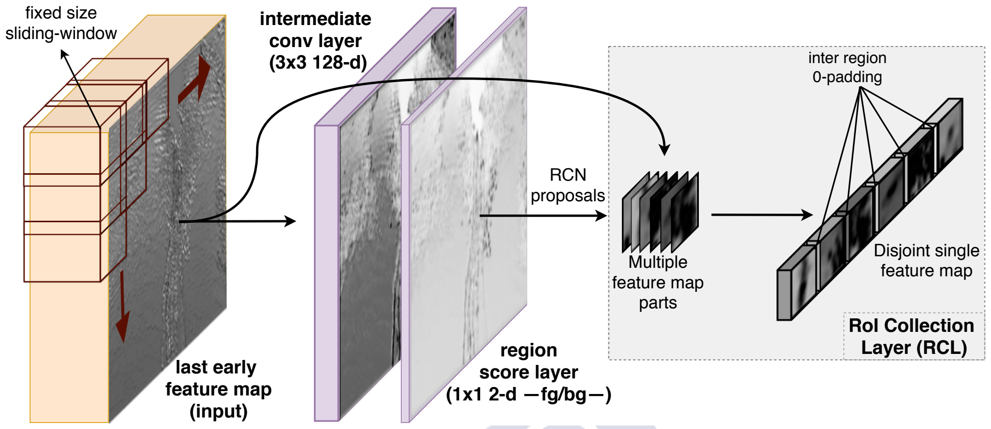

fixed size sliding-window
intermediate conv layer (3x3 128-d)
inter region 0-padding
RCN proposals
Multiple feature map parts
Disjoint single feature map
RoI Collection Layer (RCL)
region score layer (1x1 2-d —fg/bg—)
last early feature map (input)

Figure 2.4: Region Context Network (RCN) architecture.

As a summary, the key methods of the designed architecture, RCN and RCL, allow to focus all efforts on promising areas, not having to pay attention to areas without relevant information. This allows to increase the resolution of the deeper layers and, even so, reduce the use of memory and increase the frame rate. STDnet does not increase the GES to more than 4 during the whole network, while the semantic information grows. Both, high resolution and high semantic information are crucial to detect small objects.

2.3.1 Region Context Network (RCN)

The Region Context Network (RCN) is a fully convolutional network that scans a shallow convolution map in order to detect fixed-size areas where there is most likely an object. The RCN architecture is shown in Figure 2.4. RCN selects the most likely candidate regions with one or more small objects together with their context. As at this stage the goal is not to get accurate object localization, the output regions' size will be the same for all of them and neither a box regression approach, nor a set of anchors with different scales and aspect ratios are needed.

RCN consists of a first 3×3 convolutional layer over the input layer to map it into an intermediate 128-d layer with ReLU [91] following. This layer feeds a 1×1 convolutional 2-d layer to classify regions as foreground (fg), i.e., region with small objects inside, or background (bg), i.e., region without objects. RCN processes the region as a sliding-window on the feature map. The output are the scores associated with the different zones of the input image.

During the training phase, the bounding box ground truths must be grown proportionally in all directions until they reach the defined size to verify easily that the region under study represents a positive or a negative candidate. Then, as ground truths and regions have the same size, the overlap (measured by the intersection-over-union (IoU) ratio) between them is representative and can be calculated to assign each region a positive label, if IoU>0.8, or negative, if IoU<0.3. The objectness score of the candidate regions in RCN is minimized through:

$$L_{RCN}(\{p_i\}) = \underbrace{\frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*)}_{\text{fg/bg classifier}}, \tag{2.1}$$

where p_i is the predicted probability of the i-th region being foreground in an RCN mini-batch, and p_i^* is the adapted ground-truth label. The term $\frac{1}{N_{cls}}$ normalizes the equation and it refers to the size of the RCN's mini-batch. L_{cls} is a cross-entropy loss over regions with or without objects (fg/bg).

RoI Collection Layer (RCL)

The promising regions generated by RCN cannot be processed separately due to the overhead that it entails and the need to modify the remaining backbone stages –late convolutions. Instead, a novel layer is implemented where RCN ends up, the so-called RoI Collection Layer (RCL) (Figure 2.4). RCL layer takes as input the feature map generated by the last early convolution and the top scored proposals from RCN to return a single filtered feature map with the same information as that of the input feature map, but only for the set of selected regions. These regions will be concatenated in the new feature map in a disorderly way. Successive convolutions with filters greater than 1×1 will affect the neighboring regions' outputs. To solve this problem, RCL adds an inter region 0-padding —shown by gaps between regions in Figure 2.4.

With this configuration, the dimensions of the feature map output are obtained as follows:

$$RCL_{output\ size} = \underbrace{(r_w n + p_d(n-1))}_{\text{width}} \times \underbrace{r_h}_{\text{height}}, \tag{2.2}$$

where n is the number of regions from RCN, r_w and r_h are the dimensions of the regions in the RCL input feature map and p_d is the size of the 0-padding between regions. For example,

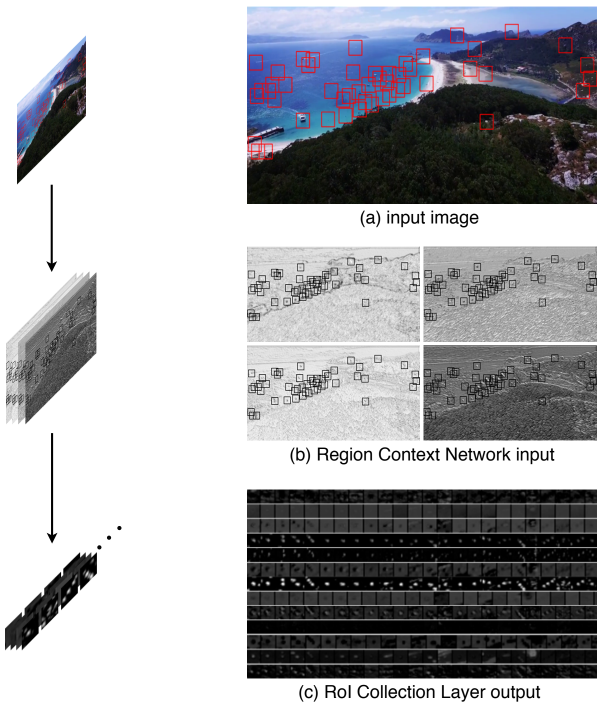

(a) input image
(b) Region Context Network input
(c) RoI Collection Layer output

Figure 2.5: An example of the feature maps involved in the RCN.

a 1280×720 input image has an RCL input feature map of 320×180, and the output RCL generates a 649×12 feature map: 50 regions of size 48×48 at the input image —12×12 at the RCL input feature map for GES = 4— with $1px$ 0-padding in the example; i.e., a reduction of 7.4× of GPU memory usage —86.5% saved memory.

Figure 2.5 shows some examples of input and output feature maps of the RCN: Figure 2.5(a) original input image and the most promising regions proposed by the RCN; Figure 2.5(b) four of the input feature maps to the RCN and the most promising regions; and Figure 2.5(c) some feature maps composed by the RCL with the most promising regions, where each row represents a different channel. Each of the feature maps in Figure 2.5(b) generates a row in Figure 2.5(c). Each row in Figure 2.5(c) represents a feature map with the promising regions separated by 0-padding.

2.3.2 Region Proposal Network (RPN)

The Region Proposal Network (RPN) used in this chapter is a modification from the one presented in Faster R-CNN [106] to deal with the feature map composed by RCL, i.e., the RPN input contains unsorted regions. In the original RPN, the anchors were processed linearly since the coordinates of its input feature map correspond with those of the input image but scaled. With the unpromising areas removed, this correspondence no longer exists and the correlation is not straightforward. RPN must take as input, besides the last feature map, the top scored promising regions information from the RCN to generate the anchors relative to those regions. Finally, the output of the bounding box regression is transformed to the input image coordinates.

Automatic anchors initialization by k-means

The approaches that rely on RPNs define the number of anchors and their sizes heuristically. In our proposal, both the number and the size of the anchors are learned through k-means. The k-means anchor learning procedure is implemented as a preprocessing stage of STDnet. k-means is applied to the training set of ground truth boxes' height and width. In order to obtain the number of kernels, which will be the number of anchors, we perform an iterative k-means with an increasing number of kernels until the maximum inter-kernels IoU exceeds a certain threshold. We have set this threshold to 0.5, which is the value used in well-known repositories, as PASCAL VOC [30] or MS COCO [80], to check if a detection is positive or negative with respect to a ground truth. This approach can be adopted by any other object detection network with anchors, e.g., Faster-R-CNN, regardless the target size of the objects.

A similar contribution was defined in [103] where a k-means algorithm selects the anchors' size according to the dataset, but where the selection of the number of anchors is done manually, visualizing the best trade-off between the number of anchors and the average intersection of these with the dataset objects. Our approach makes the anchors selection completely automatic.

2.3.3 Implementation Details

In this chapter, we adopt the approximate joint training [106] to train STDnet. To implement this end-to-end training, the ResNet-50 layers are shared with the two modules, RCN and

RPN, so that all learnable layers can be trained by backpropagation and stochastic gradient descent [73].

To train the RCN module, a mini-batch is obtained from a single input image by randomly selecting foreground and background regions. The mini-batch used within the RCN is 64 examples trying to maintain whenever possible a ratio of 1:1 of positive and negative labels. In order to eliminate overlapping regions from those proposed by the RCN, we apply an aggressive non-maximum suppression with a low threshold (0.3) over the 2,000 best proposals before the RCL, resulting in a low number of scattered regions —around 200 on average. At test, we let pass through the RCN those regions with confidence higher than 0.3, up to a maximum of 50 regions. The RCN promising regions' fixed-size was obtained estimating the effective receptive field (ERF) which, in practice, follows a Gaussian distribution [84], so half of the theoretical receptive field of the convolutions between RCN and RPN was selected as ERF. The fact that RCN and RCL modules do not alter the global batch size, makes the rest of the training identical to other two-stage networks like Faster-R-CNN. The initialization of anchors by k-means does not affect training either, since it is performed once for each new dataset and previously to STDnet training.

RCN and RCL can be theoretically integrated in any object detection framework based on ConvNets, either one-stage or two-stage approach. The main modification in addition to the new modules is to adapt the corresponding region proposal method to work with unsorted regions. In this chapter, we have implemented STDnet over Faster-R-CNN. The hyper-parameters for training and testing STDnet are the same as those used in Faster-R-CNN. The RPN module in STDnet is placed between convolutional layers C_4 and C_5 as it is done in [47] for Faster-R-CNN. Finally, at test, we apply a box-voting scheme after non-maximum suppression [38]. Our implementation uses the framework Caffe [58].

2.4 STDnet with spatio-temporal features (STDnet-bST)

STDnet detects objects using only the information coming from the current frame. Nevertheless, exploiting the information of a set of consecutive frames might help to improve detection, specially in those cases where the confidence of a detection is low. ConvNets that make detections based on a set of frames are called spatio-temporal networks [14], in contrast to conventional ConvNets —also referred to as spatial ConvNets.

STDnet with spatio-temporal features for small target detection, namely, STDnet-bST,

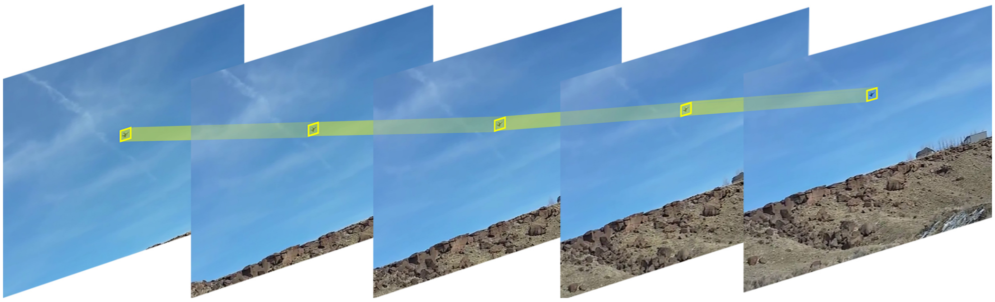


Figure 2.6: Tubelet, in yellow, generated by STDnet-bST through the Viterbi algorithm applied to the spatial detections from STDnet.

consists of two modules: (i) STDnet as a spatial detector, which provides a set of detections for the current frame; and (ii) the temporal module, which combines the detections of a set of frames and generates the final set of detections for the current frame.

The spatial module of the STDnet-bST, i.e., STDnet, feeds the temporal module with the set of detections of the current frame. The temporal module generates the data association among detections across the last τ frames through the Viterbi algorithm [41] based on the scores for all the spatio-temporal detections at the current frame, and building up what is known as tubelet. Figure 2.6 shows the composition of a single tubelet over τ different frames given a set of detections in each one. The final spatio-temporal score in STDnet-bST is estimated with an approach similar to [98], but computing the Viterbi algorithm at a *tubelet-level* object detection instead of at a video-level action detection.

The temporary score s_t^{ij} or temporal confidence between two detections d_{t-1}^i and d_t^j in two consecutive frames t and $t-1$ is given by:

$$s_t^{ij} = p_{t-1}^i + p_t^j + \lambda \cdot \mathrm{IoU}(d_{t-1}^i, d_t^j) \tag{2.3}$$

where p_t^j is the confidence returned by STDnet for d_t^j, IoU is the overlap measured as the intersection over union between d_{t-1}^i and d_t^j, and λ weighs the relevance between confidences and overlap for the final temporary score s_t^{ij}.

After calculating the score s_t^{ij} for all the combinations of detections throughout the τ frames, the Viterbi algorithm is applied to obtain the most probable sequences of detections, i.e., tubelets (V), with size τ. he Viterbi algorithm maximizes the conditional probability of the tubelets —each one represents an object seen at different time instants— given a set of

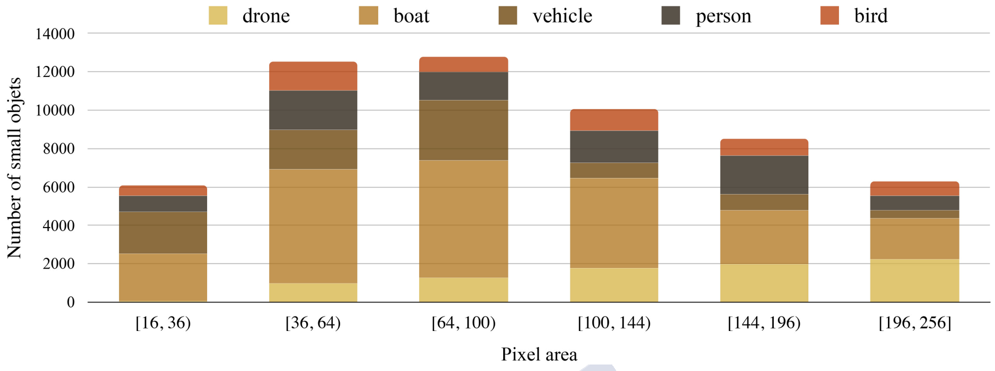


Figure 2.7: Statistics by object category and size in the USC-GRAD-STDdb database.

detections over time. The most likely tubelet ($\hat{v}$) is given by:

$$\hat{v} = \arg\max_{v \in V} \sum_{t=2}^{\tau} s_t^{i(v)j(v)} \tag{2.4}$$

where $i(v)$ and $j(v)$ are the detections at $t-1$ and t for a given tubelet $v \in V$.

The final confidence ($p_\tau^{i(\hat{v})}$) for a detection $i(v)$ at τ belonging to tubelet $\hat{v}$ will be given by the set of detections that formed the sequence $\hat{v}$, or p_τ^i if d_τ^i does not belong to a tubelet. We have experimented with several options as the mean, maximum or median of the detections' score to compute $p_\tau^{i(\hat{v})}$, being the mean the most accurate one:

$$p_\tau^{i(\hat{v})} = \frac{1}{\tau} \sum_{t=1}^{\tau} p_t^{i(\hat{v})} \tag{2.5}$$

2.5 Experiments

In this section, we release our Small Target Detection database (USC-GRAD-STDdb), and we conduct extensive experiments for our approach and previous state-of-the-art works. We also assess STDnet on the 80 category Microsoft COCO 2017 detection dataset [80]. Finally, a series of computational optimizations are made over STDnet and STDnet-bST to map them into an embedded GPU.

Area	[16,36]	(36,64]	(64,100]	(100,144]	(144,196]	(196,256]	[16,256]
# objs	6,074	12,513	12,759	10,056	8,497	6,303	56,202
% db	10,8%	22,3%	22,7%	17,9%	15,1%	11,2%	100%

Table 2.1: Statistics of the number of objects in the USC-GRAD-STDdb database according to their size.

2.5.1 The Small Target Detection database (USC-GRAD-STDdb)

The Small Target Detection database (USC-GRAD-STDdb)[2] is a set of annotated video segments retrieved from YouTube. USC-GRAD-STDdb comprises 115 video segments with more than 25,000 annotated frames of HD 720p resolution ($\approx 1280 \times 720$) with small objects of interest from 16 ($\approx 4 \times 4$) to 256 ($\approx 16 \times 16$) as pixel area. Figure 2.1a and Figure 2.2 show some samples of USC-GRAD-STDdb. The length of the videos changes from 150 up to 500 frames. The total number of labeled small objects is over 56,000.

Figure 2.7 shows a histogram of the number of objects in each category and their pixel area (see Table 2.1 for more details). Although USC-GRAD-STDdb has been generated by identifying the different categories of objects through human intervention, for the experiments carried out below, a single category of object will be used, so that the output of the STDnet is either object or background. As there are many potential small object candidates, we restrict to those targets that can potentially move, even though they can be still in a given frame or set of frames. The videos in USC-GRAD-STDdb comprise the three main landscapes with five object categories, namely: air (drone, bird), 57 videos with 12,139 frames; sea (boat), 28 videos with 7,099 frames; and land (vehicle, person), 30 videos with 6,619 frames.

In the following experiments, 80% of the videos of USC-GRAD-STDdb were used for training (92 videos), while the remaining 20% were used for testing (23 videos), keeping as much a similar ratio as possible for the three different landscapes, object sizes and categories.

2.5.2 Evaluation Metrics

USC-GRAD-STDdb has been evaluated with four different metrics for all networks under study:

[2]USC-GRAD-STDdb is publicly available under request.

- The Average Precision ($AP_{@.5}$) gives the percentage of objects correctly detected, i.e., the objects for which there is at least 50% of IoU between the detected and the ground-truth bounding boxes, averaged over categories [30].
- $AP_{@[.5,.95]}$, which is the average AP when the percentage of IoU goes from 50% to 95% in 5% steps, as reported in MS COCO [80].
- The *average number of false positives per image* (FPPI) when recall = 0.5, and the Recall for FPPI = 1 [109]. The Recall measures the ratio of true object detections to the total number of objects in the dataset for a given confidence threshold. In this case, for the confidence threshold that obtains exactly FPPI = 1.

Additionally, in the case of MS COCO, we report the COCO-style metrics, i.e., $AP_{@.5}$, $AP_{@[.5,.95]}$, $AR_{@.5}$ and $AR_{@[.5,.95]}$.

All the above metrics are calculated on the basis of precision (P) and recall (R), whose definitions are:

$$\begin{aligned} P &= \frac{TP}{TP+FP} \\ R &= \frac{TP}{TP+FN}, \end{aligned} \tag{2.6}$$

where TP stands for true positives, FP for false positives and FN for false negatives for a given IoU threshold. Then, with the output detections ranked by confidence, each one is assigned to a label (TP, FP or FN), generating a set of precision-recall points —they can be represented in a precision-recall curve as Figure 2.8(left).The Average Precision (AP) is given by finding the area under the precision-recall curve for each category and averaged over all categories. The Average Recall (AR) is the maximum recall value obtained for each category and averaged over all categories.

2.5.3 Results on USC-GRAD-STDdb

Table 2.2 through Table 2.5 show the experimental results on USC-GRAD-STDdb with the spatial network, i.e., STDnet. Our approach is compared to the state-of-the-art Faster-R-CNN [106], FPN [78] and RetinaNet [79] networks. FPN is the base of the top 3 entries of 2018 COCO object detection challenge [90]. We report all metrics described above as well as the GPU memory and frame rate during testing. The global effective stride (GES) refers to the

Method	Anchors			$AP_{@.5}$	$AP_{@[.5,.95]}$
	Scales	Aspect ratios	# anchors		
Faster-R-CNN[106]	128^2, 256^2, 512^2	1:1, 2:1, 1:2	9	19.3	5.2
Faster-R-CNN[106]	8^2, 16^2, 32^2	1:1, 2:1, 1:2	9	21.7	5.4
Faster-R-CNN[106]	4^2, 8^2, 16^2	1:1, 2:1, 1:2	9	20.3	5.4
Faster-R-CNN[106]+k	8×7, 14×10, 10×16, 21×9		4	**25.5**	**6.4**

Table 2.2: Performance of different RPN anchor scales compared to k-means on USC-GRAD-STDdb.

$RCN_{region\ size}$	STDnet-C2		STDnet-C3	
	$Recall_{RCN}$	$AP_{@.5}$	$Recall_{RCN}$	$AP_{@.5}$
32×32	91.8	54.5	93.9	**57.4**
48×48	95.2	**56.5**	96.6	57.0
64×64	95.4	55.8	96.6	56.2

Table 2.3: STDnet performance obtained by varying the size of the RCN output regions.

downscaling of the input image with respect to the feature map in the convolutional layer before the shallower RPN —C_4 for Faster-R-CNN and C_2 for FPN. Regarding RetinaNet, the original paper indicates that they do not use the high resolution feature map C_2 to locate objects for computational reasons, but the experiments on the USC-GRAD-STDdb report that starting at C_3, where GES = 8, the performance is very poor since the objects are too small. Therefore, a configuration more similar to that of FPN to locate objects has been selected.

Table 2.2 compares the performance of Faster-R-CNN with its original GES (16) for different values of the anchors of the RPN. The first row corresponds with the configuration for the Pascal VOC [106]; the anchors of the second and third rows have been adapted manually to the new database, USC-GRAD-STDdb; finally, the last row uses the anchors selected by our proposal based on k-means. The k-means learning for USC-GRAD-STDdb results in just 4 defined anchors. All the evaluation metrics with heuristic anchors are below those met with our proposal based on k-means. From these results, in the experiments that follow, the baseline Faster-R-CNN will take as anchors those defined through our k-means proposal.

Table 2.3 studies the sizes of the regions in the RCN to assess that the estimation of the ERF is half of the theoretical receptive field. For STDnet-C2, as from C_2 to C_4 there are ten 3×3 convolutions with nonlinear activations in addition to the 3×3 RPN convolution, the theoretical receptive field is 23×23 between these blocks of convolutions. For STDnet-C3,

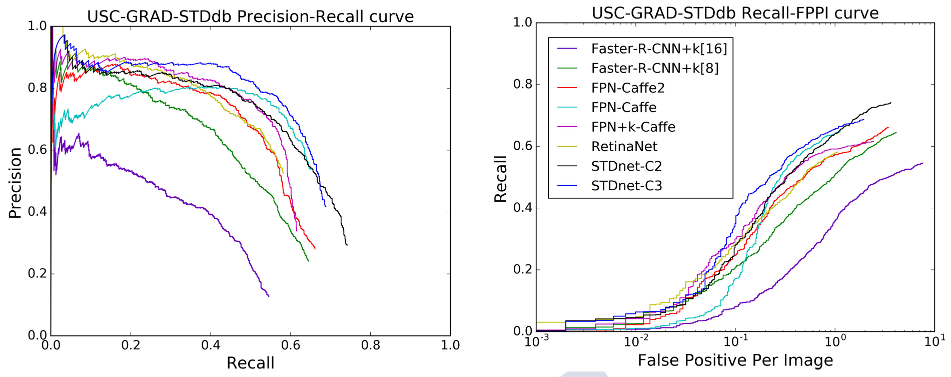


Figure 2.8: Precision-recall (left) and recall-FPPI (right) curves. The numbers inside the brackets indicate the global effective stride (GES).

Method	GES	$AP_{@.5}$	$AP_{@[.5,.95]}$	Recall	FPPI	fps	Mem.(GB)
Faster-R-CNN[106]+k	16	25.5	6.4	35.78	3.35	2.9	7.9
Faster-R-CNN[106]+k	8	44.0	14.4	50.73	0.95	2.6	10.8
Faster-R-CNN[106]+k	4	—	—	—	—	—	$>24.0_{train}$
FPN[78]	4	49.2	16.6	57.28	0.48	7.6*	2.8*
FPN	4	50.8	16.3	63.02	0.29	3.0	**6.9**
FPN+k	4	50.7	16.8	59.14	0.31	3.5	**6.9**
RetinaNet[79]	4	47.6	16.2	57.87	0.47	6.5*	3.1*
STDnet-C2	4	56.5	17.9	64.03	0.32	**4.8**	7.2
STDnet-C3	4	**57.4**	**20.0**	**65.49**	**0.22**	3.7	10.6

Table 2.4: Evaluation metrics of Faster-R-CNN, FPN, RetinaNet and STDnet on USC-GRAD-STDdb. +k indicates that the anchors were defined by the k-means algorithm. The computational entries denoted by "*" run on Caffe2 framework, so the comparison with Caffe implementations is not fair in terms of fps and memory consumption.

the theoretical receptive field is 15×15 between C_3 and C_4. Thus, regions of 48×48 —$\approx 12 \times 12$ with GES 4— and 32×32 —$\approx 8 \times 8$ with GES 4— will be used, respectively. Results support this hypothesis. Larger regions pass more true objects, increasing the recall of RCN ($Recall_{RCN}$), but with a lower AP because also more background is passed through. The key idea is that these regions should be as small as possible to preserve memory but, also, they have to contain the largest objects defined as small targets, and exploit the ERF of late convolutions between RCN and RPN.

Table 2.4 and Figure 2.8 compare the performance of Faster-R-CNN, FPN, RetinaNet and

STDnet on USC-GRAD-STDdb. It also shows the effect of placing the RCN in STDnet after the second or third convolutional blocks, namely, STDnet-C2 and STDnet-C3, respectively. The deeper the RCN, the better the evaluation metrics, but at the cost of more memory usage and less frame rate.

For Faster-R-CNN, as expected, finer effective strides lead to better metrics. GES = 4 in the baseline Faster-R-CNN exceeds the size of our GPU memory at training. The STDnet allows to work with lower GES, outperforming Faster-R-CNN in $AP_{@.5}$ —57.4% vs. 44.0%— and $AP_{@[.5,.95]}$ —20.0% vs. 14.4%. STDnet also improves the FPPI $4.3\times$ and the speed rate $1.4\times$.

When it comes to the FPN, the performance of two different implementations have been reported on USC-GRAD-STDdb. FPN [78] runs on Caffe2, in the same repository as RetinaNet[3], and they take advantage of its speed performance and memory optimization improvements. FPN (Caffe) is programmed in Caffe framework, starting from the Faster-R-CNN official code[4] —just like STDnet. FPN in Caffe and Caffe2 provide similar results in AP, as seen in Table 2.4.

The architecture of FPN presents RPNs at different scales —with the first one at the C_2 level with GES = 4—, which leads to higher performance than Faster-R-CNN, reaching $50.8\%_{@.5}$ and $16.3\%_{@[.5,.95]}$. Nonetheless, the upsampling performed from the deepest convolutions causes a lower performance compared to STDnet, which reaches $57.4\%_{@.5}$ and $20.0\%_{@[.5,.95]}$. Moreover, the FPPI is $1.3\times$ better for STDnet, which is also $1.2\times$ faster. During the experimentation, we tested two configurations for the combination of FPN and k-means: (i) the same anchors at each RPN; and (ii) the set of anchors distributed among the different RPNs. The best results for FPN+k (shown in Table 2.4) were obtained with the second option. Comparing FPN and FPN+k, the performance of FPN+k for $AP_{@.5}$ is slightly worse than FPN, because the first RPN in FPN already has a set of anchors suitable for small objects. Nevertheless, for $AP_{@[.5,.95]}$, FPN+k improves FPN by a 0.5%, which indicates that the k-means algorithm helps to generate better bounding boxes.

RetinaNet works with GES = 4 as FPN, and it obtains competitive results using a one-stage approach. Nevertheless, the need to place the shallowest set of anchors in C_2 lowers both the accuracy and the computational performance [79].

Finally, Table 2.5 shows the results for different object sizes of small targets as defined

[3]https://github.com/facebookresearch/Detectron
[4]https://github.com/rbgirshick/py-faster-rcnn

AP_{IoU}	Method	[16,36]	(36,64]	(64,100]	(100,144]	(144,196]	(196,256]
@.5	FPN	22,74	48,70	**65,48**	71,24	61,22	65,12
	STDnet-C3	**27,14**	**53,50**	61,33	**76,88**	**69,83**	**76,06**
@[.5,.95]	FPN	6,48	11,89	**21,69**	23,75	22,19	20,60
	STDnet-C3	**7,79**	**14,17**	20,93	**26,37**	**29,41**	**31,06**

Table 2.5: STDnet and FPN performances for different object sizes (area in pixels) of the USC-GRAD-STDdb database.

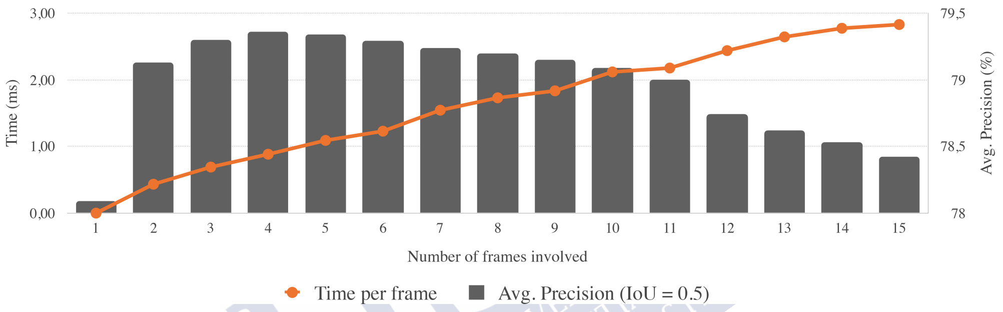


Figure 2.9: Viterbi algorithm runtimes and the average precision obtained for different values of τ in the USC-GRAD-STDdb training set.

in Section 2.5.1. As expected, the smaller the size of the objects, the lower the performance. STDnet outperforms FPN in 5 out of the 6 object sizes for both $AP_{@.5}$ and $AP_{@[.5,.95]}$ metrics. We highlight that STDnet is over 20% in $AP_{@[.5,.95]}$ for most of the object segments. $AP_{@[.5,.95]}$ is a very meaningful metric as it encompasses AP as IoU reaches perfection.

As addressed in Section 2.4, STDnet-bST includes temporal features through tubelets built up with the Viterbi algorithm. To determine the STDnet-bST hyperparameters, we used the USC-GRAD-STDdb training set. The impact of the number of frames (τ) is analyzed in Figure 2.9 which shows a comparison between the time needed to process the temporal stage for each frame and the AP for different number of frames. $\tau = 4$ presents a good trade-off between computation time and accuracy.

To sum up the achieved performance, Table 2.6 shows a comparison between STDnet and STDnet-bST. As seen, STDnet-bST outperforms STDnet in all metrics. The processing time is practically identical, reaching 3.7 fps, since the overhead added by the temporal module to the STDnet is negligible.

Method	$AP_{@.5}$	$AP_{@[.5,.95]}$	Recall	FPPI
STDnet	57.4	20.0	65.49	0.22
STDnet-bST	**59.7**	**20.6**	**66.81**	**0.20**

Table 2.6: Performance of STDnet-bST compared to STDnet over USC-GRAD-STDdb database.

Method	$AP^{xs}_{@.5}$	$AP^{xs}_{@[.5,.95]}$	$AR^{xs}_{@.5}$	$AR^{xs}_{@[.5,.95]}$
Faster-R-CNN	5.0	1.5	22.0	7.6
FPN	**11.8**	4.8	**36.7**	15.9
RetinaNet	9.1	4.5	33.0	16.2
STDnet-C3	11.4	**5.5**	36.0	**17.3**

Table 2.7: Evaluation metrics of Faster-R-CNN, FPN, RetinaNet and STDnet on the *extrasmall* objects of MS COCO val 2017 subset, i.e., objects under 256 pixels of area.

2.5.4 Results on MS COCO 2017

MS COCO [80] is a popular image dataset for object detection with 108,556 small objects defined as those objects with an area of less than 32 $\times$ 32 pixels. We have defined a new scale subset —*extrasmall* (AP^{xs})— within the category small objects of COCO to include small targets as defined in this chapter, i.e., those enclosed in bounding boxes with less or equal than 256 pixels of area —not the segmentation area as in the original annotations. As we have defined our own subset, we cannot evaluate the results with the official COCO test-dev 2017. Instead, we train with COCO train 2017 and evaluate with the popularly extended COCO val 2017 (5k) [4]. Considering this, the total amount data used is: 62,658 of 236,574 objects from COCO train 2017 and 2,562 of 10,000 objects from COCO val 2017.

There are a few minor changes that should be made in STDnet for this dataset. The images are re-scaled such that their shortest side is 600 as in the baseline Faster R-CNN [106]. Also, the RCN output regions have a size of 64$\times$64 due to both the re-scaling and the extremely elongated nature of some categories —such as book, or skateboard. To conclude, we work with a mini-batch size of 128 regions to train RCN.

Table 2.7 shows the performance comparison between our approach, its baseline Faster-R-CNN, FPN and RetinaNet. Here, we report the MS COCO evaluation metrics AP^{xs} and AR^{xs} for the *extrasmall* subset. In the same way as in the USC-GRAD-STDdb, the STDnet obtains much better results than its baseline Faster-R-CNN, improving $AP_{@.5}$ by more than 2$\times$ and $AP_{@[.5,.95]}$ by 3$\times$, while surpassing by 14.0% and 9.0% in $AR_{@.5}$ and $AR_{@[.5,.95]}$, respectively. Comparing STDnet with the state-of-the-art FPN, it is observed how the detections provided by STDnet suit better to the ground truth, yielding 0.7% ($AP_{@[.5,.95]}$) and 1.4% ($AR_{@[.5,.95]}$) higher than FPN detections. This metric is considered the most important —primary challenge metric— by MS COCO [80] because it encompasses AP adding information on how it behaves as the IoU reaches perfection. Finally, as expected, RetinaNet obtains lower performance than FPN and STDnet due to the same reasons mentioned for the dataset USC-GRAD-STDdb. In addition, the MS COCO presents objects very close to each other, which causes a disadvantage for the one-stage approaches.

It should be noted that the object detection MS COCO dataset, despite being the most complete and used repository in the field, features some issues when we refer to extremely small objects, which affects performance metrics.

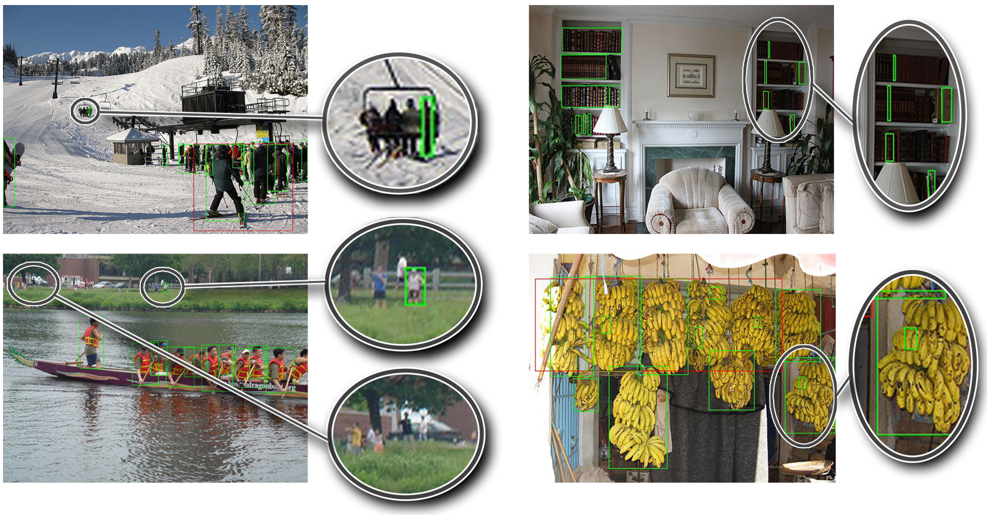
The first issue is the lack of annotations when a large number of objects of the same class are grouped. Some of these occurrences are solved with the *iscrowd* label in the annotation, but in some others this label does not exist or, if it does, it is incorrect. Some examples are displayed on Figure 2.10a, where COCO annotations are shown.

The second issue is the existence of parts of large objects labeled as small objects with an *extrasmall* size for being largely occluded. Some of these examples are shown in Figure 2.10b. This poses a challenge for any detection algorithm. Nevertheless, our approach suffers more from this issue than FPN since STDnet features a receptive field considerably smaller than that of the FPN, as the size of the feature maps do not change in STDnet after passing through the RCN.

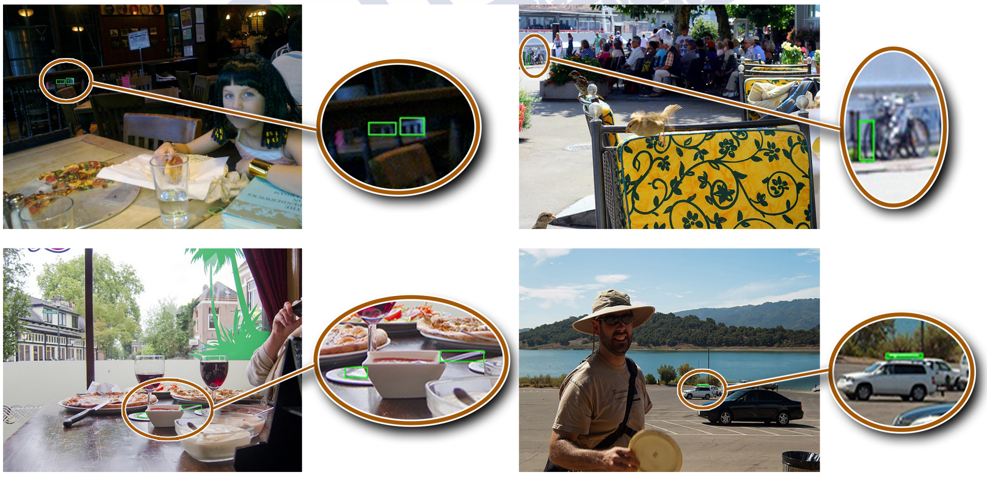
2.5.5 Execution on Embedded GPUs

Embedded GPUs are oriented to on-board platforms and, as such, they feature a limited computing capacity when compared to their GPU desktop counterpart. Therefore, it is necessary to reduce memory consumption and to improve computation time to migrate STDnet and STDnet-bST to embedded GPUs. The optimization carried out in this work is based on the unification of the contiguous blocks of convolution and batch normalization methods at test stage [51].

The description of this merger method uses the Caffe's notation [58]. The batch normalization step from [56] also included a per-channel learned bias and scaling factor so, in Caffe's implementation, it is splitted into two layers named batch normalization and scale layers with the following parameters.



(a) Images with non-labeled objects or incorrect labels where only the category of interest is marked.



(b) Images with objects parts with extrasmall size where only the category of interest with extrasmall size is marked.

Figure 2.10: Some examples of controversial small labels on MS COCO val 2017. Normal objects are colored green and *iscrowd* objects are colored red (best seen in color).

- Convolution layer: convolutional weights (c_w) and convolutional bias (c_b).
- Batch Normalization layer: global mean (bn_{mean}), global variance (bn_{var}) and moving average factor (bn_{norm}).
- Scale layer: scaling factor (s_w) and per-channel bias (s_b).

Considering the above notation, the three layers —convolution, batch normalization and scale— can be unified at test stage without altering the final result as follows:

1. A β vector is computed as a multiplier factor for the convolutional data:

$$\beta = \frac{s_w}{\sqrt{\frac{bn_{var}}{bn_{norm}+\varepsilon}}} \tag{2.7}$$

   where ε is a small value added to the variance estimate to avoid division by zero.

2. The convolutional weights c_w and bias c_b trained can be updated as:

$$\begin{gathered} c_w(i) = \beta(i)c_w(i) \\ c_b = \beta c_b + \left(s_b - \beta\frac{bn_{mean}}{bn_{norm}}\right) \end{gathered} \tag{2.8}$$

3. The batch normalization and scale layers can be bypassed by resetting their parameters as default, which is the same as removing those layers.

Table 2.8 shows the performance of the optimizations on HD 720p images for the two versions of STDnet implemented, with the RCN layer after the two and three convolution blocks of the network (STDnet-C2 and STDnet-C3, respectively). These metrics have been measured on a high-performance cluster GPU. The memory is reduced by 61.4% for the STDnet-C2 version and by 65.5% for STDnet-C3, in addition to improving the computation time by 52.2% for STDnet-C2 and 40.9% for STDnet-C3.

The NVIDIA Jetson TX2 [35] has been selected as an embedded and portable device to perform the tests. This architecture has a middle-low graphic card (NVIDIA Pascal with 256 cores) and a limited memory of 8GB shared between the CPU and the GPU. The performance on this device has been evaluated both for HD 720p images —the originals of the USC-GRAD-STDdb database— and VGA images. In the case of VGA images, to perform the

Method	STDnet-C2		STDnet-C3	
	mem. (GB)	fps	mem. (GB)	fps
STDnet	7.22	4.80	10.61	3.73
STDnet opt.	4.29	5.75	5.95	4.59

Table 2.8: Comparison of STDnet and the optimized version of STDnet in memory consumption and computation time (fps).

Method	VGA		HD 720p	
	mem. (GB)	fps	mem. (GB)	fps
STDnet-C2	4.76	1.41	5.19	0.77
STDnet-C3	4.84	1.23	5.98	0.59

Table 2.9: Performance in memory consumption and computation time, shown in frames per second (fps), over the Jetson TX2 architecture for VGA and HD 720p images.

tests, segments of size 640×480 were selected from videos in the database that contained some small objects. The computational performance results on Jetson TX2 are shown in Table 2.9. As seen, the memory used differs slightly from that of a cluster GPU due to their different memory management procedures.

2.6 Conclusions

We have introduced STDnet, a region-proposal-based ConvNet to detect small targets under 16×16 pixels. The key of STDnet is an additional visual attention mechanism that we call RCN that chooses the most likely candidate regions with one or more small objects and their context. RCN allows for finer effective strides that lead to greater precision while saving memory usage and increasing frame rate. We have also included an automatic definition of the anchors with k-means that improves the classical heuristic approach.

In addition, we have released a new video dataset, USC-GRAD-STDdb, with more than 56,000 annotated small objects in complex backgrounds with clutter. STDnet obtains the best results on USC-GRAD-STDdb with a 57.4% $\text{AP}_{@.5}$ and 20.0% $\text{AP}_{@[.5,.95]}$, clearly outperforming its counterparts. STDnet-bST even improves these results without adding overhead, from 57.4% $\text{AP}_{@.5}$ to 59.7% $\text{AP}_{@.5}$. Furthermore, we have tested our approach with the *extrasmall* objects that exist in MS COCO, where the overall performance of STDnet is very competitive.

Finally, we have deployed STDnet and STDnet-bST on an embedded GPU, the Jetson TX2. For that, we have implemented a number of optimizations that allow to run both ConvNets on Jetson TX2 by reducing the consumed memory by 59% and increasing the fps by 20%.

Chapter 3

STDnet-ST: Spatio-Temporal ConvNet for Small Object Detection

In Chapter 2 we detailed STDnet for small object detection in images. In this chapter we explore the potential of the temporal features in video datasets to improve small object detection. The temporal coherence of the same object in previous frames can contribute to optimize the detection accuracy in the current frame. Specifically, this chapter introduces STDnet-ST, an end-to-end spatio-temporal convolutional neural network for small object detection in video. This approach improves the overall small object detection accuracy by using the spatial information in STDnet operating alongside temporal video information. For this purpose, STDnet-ST correlates pairs of the top-ranked RCN regions with the highest likelihood of containing small objects to find similarity and to be able to associate them. This additional level of abstraction, allows to link the small objects over time, generating tubelets. Furthermore, we propose a procedure to dismiss unprofitable object links in order to provide only high quality tubelets.

The contents of this chapter are extracted from the following publications:

> B. Bosquet, M. Mucientes, and V. M. Brea. Correlation-based ConvNet for Small Object Detection in Videos. *In Proceedings of the 25th International Conference on Pattern Recognition (ICPR)*, Milan (Italy), 2021.

B. Bosquet, M. Mucientes, and V. M. Brea. STDnet-ST: Spatio-Temporal ConvNet for Small Object Detection. Article submitted to Pattern Recognition in March 2020 and currently under second round review.

3.1 Introduction

Over the last years, the scope of object detection has witnessed significant progress [43]. Most of the state-of-the-art methods share a similar two-stage structure adopting the Faster R-CNN [106] approach, where a deep convolutional neural network (ConvNet) backbone is firstly applied to generate a set of feature maps over the whole input image followed by a detection-specific network [20, 47, 79] that provides the detection results from the feature maps.

Small object detection, typically defined as objects with a size below 32 $\times$ 32 pixels in widely adopted image datasets as MS COCO [80], is progressively gaining more interest in the scientific community [2, 9, 147]. This permits to tackle practical applications as sense and avoid on board of Unmanned Aerial Vehicles (UAVs), or video surveillance tasks where early actions are required. Small object detection accuracy lags behind that of larger objects [82], which opens the way for more improvement. This is in part due to the lack of specific architectures and datasets, with the exception of face detection, where objects are usually of small size, which makes up a field of interest by itself [2, 149].

The lack of specific datasets with small objects has been partially addressed with the rise of UAVs with built-in cameras to record wide areas in the wild with small objects and decent quality. In particular, UAVDT [27], VisDrone2019-VID [153] and, especially, USC-GRAD-STDdb [9] are video datasets with a large percentage of small objects.

Video object detection has had a recent upturn with the advent of ImageNet video object detection challenge (VID) [110], leading to spatio-temporal ConvNets [14]. These networks have been tried to exploit the richer information from several frames when compared to static images. Linking the same objects across video to form sequences, or tubelets, to improve the classification score has proved to be the most efficient technique [32, 62, 121] among the different ways to tackle this issue [16, 23, 63, 154].

This chapter addresses small object detection with STDnet-ST, a novel spatio-temporal convolutional neural network aimed at video small object detection. STDnet-ST is built on

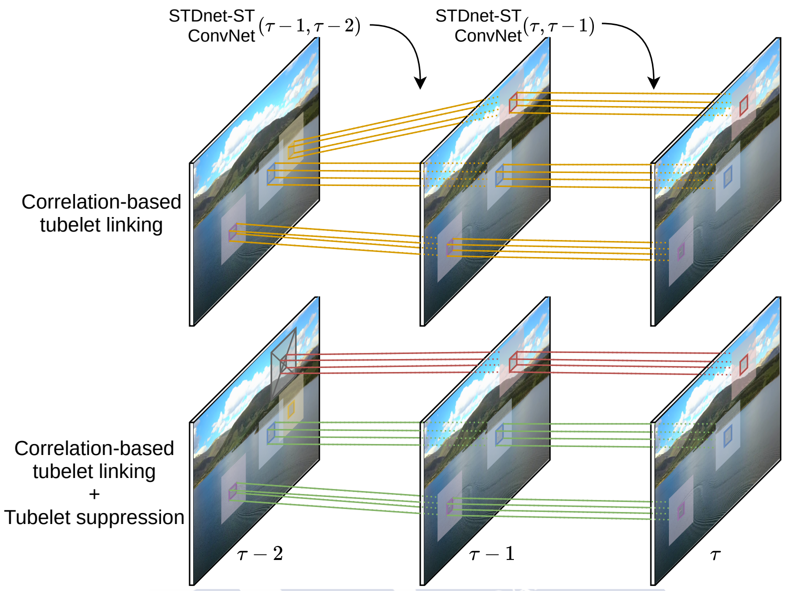

STDnet-ST ConvNet $(\tau-1, \tau-2)$
STDnet-ST ConvNet $(\tau, \tau-1)$
Correlation-based tubelet linking
Correlation-based tubelet linking + Tubelet suppression
$\tau-2$
$\tau-1$
τ

Figure 3.1: STDnet-ST has two components: *STDnet-ST ConvNet* and *STDnet-ST tubelet linking*. *STDnet-ST ConvNet* performs small object detection and correlation over two consecutive frames. *STDnet-ST tubelet linking* creates tubelets in two stages: first, the correlation-based tubelet linking creates tubelets (orange) across the last τ frames; then, tubelet suppression, generates additional nodes (⊠) to avoid unprofitable tubelets (red) while providing high quality ones (green).

STDnet [9]. STDnet is a fully convolutional neural network which provides the most likely areas of the image with small objects. Once the most promising areas with objects are selected, the rest of the image is dismissed, allowing to keep high level of detail in those selected areas without affecting the computational performance. In this chapter, we define small objects as any potentially moving object of less than 16×16 pixels without definitive visual cues to assign them to a category, following our previous work [9].

The main contributions of this work are (Figure 3.1):

- STDnet-ST, a spatio-temporal neural network built on STDnet for small object detection that operates with two input frames simultaneously. Both inputs are integrated together through a correlation module at shallower layers and a final tubelet linking.
  - The spatio-temporal ConvNet simultaneously generates the detections of the cur-

rent frame, together with the correlations between the current and previous frames. The correlation is performed in a natural way over the most promising regions of the image, i.e., regions provided by the shallowest layers of our network with a high likelihood of having objects.

– The tubelet linking is based on the Viterbi algorithm, but we include three novelties. First, it uses the correlations generated by the ConvNet to link the objects of the tubelet. Second, it scores the associations between the objects, taking into account the confidence variability of the tubelet, which is an indicator of the tubelet confidence. Third, the tubelet suppression algorithm avoids unprofitable tubelets. This is achieved by inserting additional nodes to each frame in the Viterbi algorithm based on the information coming from promising areas without detections. All these contributions allow STDnet-ST to increase the confidence of the detections most likely to be true positives within high quality tubelets and decrease the confidence of those most likely to be false positives within unprofitable tubelets.

- STDnet-ST achieves state-of-the-art results for small object detection on the publicly available datasets USC-GRAD-STDdb, UAVDT and VisDrone2019-VID, over the extremely small object subset *XS* ($\leq$ 256 px), defined in [9].

3.2 Related Work

The image object detection scope has followed two parallel trends: region proposal based detectors (*two-stages*), according to the milestone set by Faster R-CNN [106], and detectors that directly predict boxes from feature maps (*one-shot* or *one-stage*), with SSD [83] and YOLO [103] as pioneers. A large number of outstanding improvements have been derived from these architectures, being the two-stage Feature Pyramid Network (FPN) [78] noteworthy since it remains as the baseline of the leading solutions in the COCO object detection challenge[1].

The trend in small image object detection is to work on data from as fine a feature map as possible, where small objects still have distinctive features. In this line, FPN's success in the small object subset of MS COCO is mainly based on merging feature maps at different scales with a Region Proposal Network (RPN) per scale [106]. Here, the coarsest RPN makes use of a shallow feature map with stride 4, preserving fine details. In contrast, architectures like Faster R-CNN present stride 16 as a starting point to seek for objects, which might not

[1]http://cocodataset.org/#detection-leaderboard (Accessed: 2020-02-10)

suffice for a good accuracy in small object detection. Following the same idea, RetinaNet [79] is an FPN-based architecture that removes RPNs and adds two subnetworks —class subnet and bounding box subnet— to detect objects in one-stage, including small objects. The main improvement is obtained through a novel loss function (Focal Loss) to address the class imbalance in one-shot detectors. Recently, [145] studies how to optimize the feature map multi-scale integration by using *gates* to extract only useful semantic information, resulting in a more effective feature map for object detection.

Similarly, our previous approach, STDnet [9], is a ConvNet for image object detection able to keep a low stride of 4 from shallow layers. The key point is the retrieval of the top-ranked regions with more likelihood of containing small objects from shallow layers of the network. This allows to dismiss the remaining part of the input image without affecting the final accuracy while keeping a reasonable computing time.

As another approach, MDFN [85] is a recent one-shot ConvNet that proposes only to exploit high-layers and, at the same time, improve the small and occluded object detection. This is done by introducing inception modules with multi-scale filters to enhance both the semantic and contextual information. Here, as in STDnet, it is shown that context is quite relevant for detecting small objects.

Another promising research direction is based on boosting the scarce features of small objects using super-resolution (SR) techniques. On the one hand, this can be achieved by increasing the resolution of the whole input image [52], but it affects the computing time considerably. On the other hand, this can be handled by focusing only on the areas where there are small objects and applying there SR techniques. As an example, Noh *et al.* [93] propose a SR feature generator based on a GAN that learns to augment the features under the guidance of a SR feature discriminator.

Concerning the refinement of the final bounding box, there are solutions built on existing two-stage architectures that add additional headers to the existing one. These additional headers can be composed of the last convolution blocks [9, 47] or of fully-connected layers [20, 78, 106]. Finally, a classifier assigns a category and a bounding-box regressor applies a final regression for each proposal. In this line, various studies have attempted to improve the quality of the final header. Gidaris and Komodakis [38] replicate the regressor stage to refine the bounding-box iteratively. In [146], they combine various classifiers trained with the integral loss. Similar ideas are exploited in [13] to build Cascade R-CNN, improving two-stages detectors by applying consecutive headers trained with different proposals so that each one is fed by the previous.

In [128], to also address the inaccurate localization, they propose a hierarchical objectness network (HON) that refines the candidate proposals by what they call *stripe objectness*, which computes the in-out objectness and border objectness, instead of regressing the coordinates. With a similar purpose, Tao *et al.* [122] introduce a Focused Attention (FA) mechanism along with a class aware RPN (CARPN) which uses a new strategy for anchor generation that covers all scales but with fewer anchors to considerably reduce false positive proposals.

Video object detection has been widely studied for the last few years [32, 62, 121]. Several methods have been re-adapted from successful architectures in action detection [41, 97, 115]. Two-stream ConvNets are spatio-temporal networks that have achieved remarkable results [115]. The two-stream method has been studied by [97], where a Faster R-CNN has two RPNs operating over two streams of spatial and motion information from stacking optical flow over several frames.

Concerning video object detection, the solution addressed in [32] builds on R-FCN [20] with a correlation operator inserted between two input frames to extract motion information of the objects across time. The correlation operates over the entire feature maps at different scales and estimates local feature similarity for various offsets between the two frames. Then, they link the detected objects into tubelets and reweight the detections' scores within them. Correlating whole feature maps implies that, as an object becomes smaller, their movement represents a considerably smaller influence, even though the correlation acts on several scales.

The approach in [63] performs video object detection in the current frame and tracks the objects through neighboring frames in order to modify their original detections for higher accuracy. The linking among detections in different frames is based on the mean optical flow vector within boxes. Similarly, the approach in [62] links objects into long tubelets using a tracking algorithm and then adopts a classifier to aggregate the detection scores in the tubelets.

Another alternative for video object detection introduced in [121] proposes a modified RPN called Cuboid Proposal Network (CPN) for detecting objects in multiple input frames. The cuboid proposals are regressed and classified to create short tubelets. Consecutive short tubelets are merged into long tubelets by a linking algorithm that takes the best detection for each overlapping frame between two tubelets.

In Flow-Guided Feature Aggregation (FGFA) [154], authors aggregate spatial features over time based on feature correspondences computed by optical flow to improve detections. Deng *et al.* present Relation Distillation Networks (RDN) [23], which aggregate and propagate object relation using the region proposals of current and neighboring frames to enhance the features of each object proposal, and thus capturing the core features of a given object across a video. In [16], authors introduce Memory Enhanced Global-local Aggregation (MEGA), a spatio-temporal ConvNet that relies on a novel Long Range Memory (LRM) module to efficiently aggregate global and local information from key frames. MEGA achieves state-of-the-art result (85.4% mAP) on ImageNet VID dataset [110].

The spatio-temporal ConvNet for video object detection we present in this chapter, STDnet-ST, is built on our previous network, STDnet [9], which aimed at image object detection. STDnet-ST works on two consecutive frames. The retrieval of a fixed number of the top-ranked regions with more likelihood of containing small objects by the underlying STDnet eases the spatio-temporal procedure of STDnet-ST. In fact, as a difference with previous correlation-based solutions like [32], which runs correlation on the whole feature maps, STDnet-ST correlates pairs of regions with a high likelihood of having objects inside. This is a key point for small object detection, as the influence of the objects in the correlations calculated for the whole feature maps decreases with the size of the objects themselves — correlation values are mostly due to the background. Estimating the correlation for specific regions of the image allows to obtain correlation values influenced by the objects. This, in turn, permits to process only high quality tubelets by linking the objects inside such regions, which increases accuracy.

3.3 STDnet-ST Architecture

STDnet-ST is a spatio-temporal convolutional neural network for the detection of small objects in video, i.e., objects smaller than 16×16 as defined in this chapter. STDnet-ST has two components:

- The spatio-temporal convolutional neural network, which takes as inputs the current (f_t) and previous (f_{t-1}) frames, and returns the set of detections ($\mathscr{D}_t$), their confidences ($\mathscr{P}_t$), and the correlations (Sec. 3.3.1) among the detections at t and $t-1$ (C_t). These correlations will be used to associate the detections of both time instants.

- The STDnet-ST tubelet linking, which is based on the Viterbi algorithm, and includes the correlation-based tubelet linking and the tubelet suppression procedure.

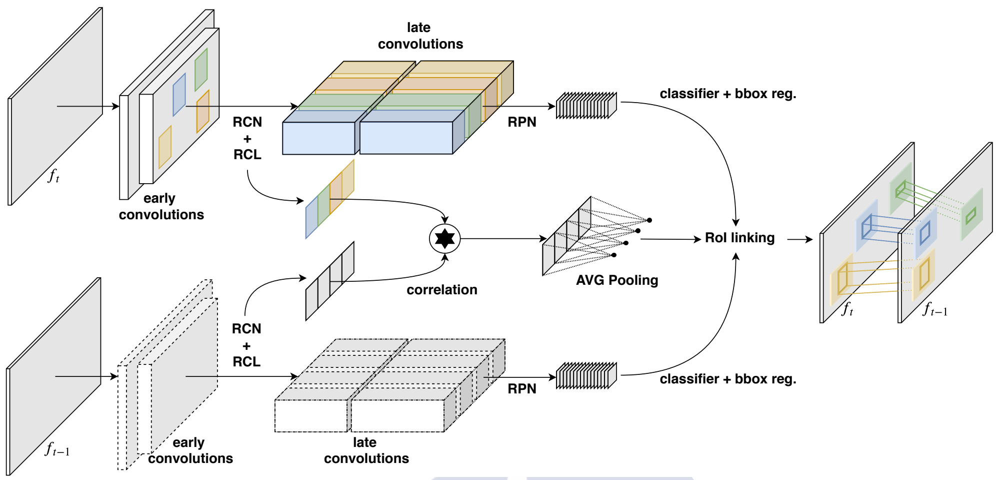


Figure 3.2: STDnet-ST ConvNet architecture. Each branch performs RCN+RCL to obtain the most promising regions (RCN regions) that are further refined into detections by the RPN and a classifier. Simultaneously, the two sets of RCN regions feed a correlation module that associates the correlation values to the final detections.

- The correlation-based tubelet linking (Sec. 3.3.2), that links the detections obtained at different time instants ($t = 1, \dots, \tau$), generating the optimal tubelets along time for each of the objects. The final goal of tubelet linking is to update the scores of the detections at time τ using the previous $\tau - 1$ detections, according to the confidence of the whole tubelet (Sec. 3.3.2). A key element of the tubelet linking is the correlation provided by the spatio-temporal ConvNet, which evaluates the likelihood of the association of two detections. Also, the scores are updated taking into account the confidence variability of the tubelet, which indicates the confidence of the whole tubelet.
- The tubelet suppression algorithm, that filters the tubelets obtained by the correlation-based tubelet linking, eliminating those that contain incorrect data associations (Sec. 3.3.2).

3.3.1 Spatio-temporal ConvNet

Figure 3.2 shows the architecture of STDnet-ST, which consists of two sibling branches together with a correlation operation among selected regions. Each of the branches is based on the STDnet architecture [9], which is focused on the detection of small objects in images, i.e., it does not take into account temporal information.

The ability of STDnet to detect small objects is due to the high resolution of the deeper feature maps of the ConvNet. This high resolution of the last feature maps is possible because STDnet provides the most promising regions of the image in the early stages, thus focusing only on those regions that most likely contain small objects. The main components of STDnet are the following —for a more detailed description refer to [9]:

- Early convolutions. In the shallower convolutional layers, STDnet learns simple features from the objects of interest.
- Region Context Network (RCN). Just after the shallower convolutions, STDnet applies a novel detector of promising areas over the last feature map to select those regions that most likely contain small objects. Then, the m_t top scored regions $\mathscr{R}_t = \{r_t^1, \dots, r_t^{m_t}\}$ are gathered in a single feature map by the RoI Collection Layer (RCL). There are two main differences between RCN+RCL and a typical RPN (Region Proposal Network): (i) RCN returns always regions of a fixed size that contain at least an object centering in it, while RPN returns bounding boxes of objects; (ii) RCL generates a new synthetic feature map, meaning that two neighboring pixels in the feature map that belong to two different regions are not neighboring pixels in the original image.
- In the late convolutions stage, the feature maps have a high resolution due to the memory saved by ruling out non promising areas. As the output of the RCL is a feature map with disjoint areas, all convolutions are designed to keep the features of each region separated from each other through padding.
- STDnet has a single RPN that takes as input the fourth convolutional block (*conv4*), which contains the most promising areas provided by the RCN but with richer semantic information.
- The last stage of STDnet refines the outputs of the RPN, generating the final bounding boxes and classifications of the objects.

STDnet-ST works with two consecutive video frames, t and $t-1$. Both branches —based on STDnet— share the same weights throughout the execution. Each of the branches generates a set of detections ($\mathscr{D}_t$) and their corresponding confidences ($\mathscr{P}_t$). As seen in Figure 3.2, the two branches are connected through the correlation module. The correlation assesses the degree of matching between a pair of RCN regions at t and $t-1$, in order to link the final detections provided by the RPN thereafter. The hypothesis on which it relies is that each RCN region specializes in detecting a single object centered in it, allowing a straightforward extension over two detections. The correlation module consists of the two region composed feature maps generated by RCL for each branch, a correlation operator, an average pooling and a final RoI linking operation. The operation of the correlation module is as follows:

1. First, it calculates the correlation for each pair of RCN regions $< r^i_{t-1}, r^j_t >$, where $r^i_{t-1} \in R_{t-1}$, $r^j_t \in R_t$, $i = 1,\ldots,m_{t-1}$, and $j = 1,\ldots,m_t$. The output is a correlated feature map with the same width and height as the input regions, and where each pixel is obtained as the dot product of the pixels placed at that position in both regions — the depth of the correlated feature map is a single channel, due to the dot product. The correlation operator will produce a feature map with $m_{t-1} \times m_t$ regions, each one representing the correlation between two of the RCN regions.

2. Then, an average pooling is applied to summarize each of the regions of the correlated feature map in a single value associated to each pair of RCN regions, generating $m_{t-1} \times m_t$ correlation scores.

3. Finally, the correlation scores of each pair of RCN regions are associated to the final detections by the RoI linking operation. The RoI linking operation takes as input the final detections ($\mathscr{D}_t$) —generated by the RPN and further refined by the classifier— for each STDnet-ST branch, as well as the correlation scores, and outputs the correlation scores but associated to each pair of final detections, generating the matrix $\mathscr{C}_t$. $\mathscr{C}_t$ has a size of $n_{t-1} \times n_t$, where n_{t-1} and n_t are respectively the number of detections at times $t-1$ ($\mathscr{D}_{t-1}$) and t ($\mathscr{D}_t$). Those correlation scores not included in $\mathscr{C}_t$ —not all RCN regions have an associated final detection— are kept as they are involved in the tubelet suppression algorithm (Sec. 3.3.2).

3.3.2 STDnet-ST tubelet linking

The object linking across a sequence of frames to build tubelets is a popular approach to combine the temporal information. The final goal of this stage is to increase the confidence of those detections that have a high likelihood of being true positives and to reduce the confidence of those detections with a low likelihood of being true positives.

First, we describe a baseline tubelet linking approach based on the spatial overlap between boxes in neighboring frames without considering the motion information. Then, we present the STDnet-ST tubelet linking with its two components: (i) the correlation-based tubelet linking (Sec. 3.3.2), which is based on the correlation scores generated by the ConvNet (Sec. 3.3.1); and (ii) the tubelet suppression procedure (Sec. 3.3.2) that removes those unlikely tubelets retrieved from the correlation-based tubelet linking.

Baseline tubelet linking

The baseline tubelet linking is based on [97], although they apply it to action detection in video, while we use it for spatio-temporal object detection. Given a set of τ frames, first, the tubelet linking calculates the set of scores between pairs of detections in two consecutive time instants ($\mathscr{S}_t$). Then, it applies the Viterbi algorithm [41] to find the most likely sequences, i.e., tubelets ($\mathscr{V}$), for all detections in the τ frames. Finally, it recalculates the score of each detection in τ ($\hat{\mathscr{P}}_\tau$) given the tubelet it belongs to.

The first step of tubelet linking calculates the score matrix $\mathscr{S}_t = \{s_t^{11}, \dots, s_t^{n_{t-1}n_t}\}$, where s_t^{ij} is the score between two equal category detections d_{t-1}^i and d_t^j in two consecutive frames, and is given by:

$$s_t^{ij} = p_{t-1}^i + p_t^j + \lambda \cdot \text{IoU}(d_{t-1}^i, d_t^j) \tag{3.1}$$

where p_t^j is the confidence returned by STDnet-ST for the j-th detection at frame t, IoU is the overlap —measured as the intersection over union— between two detections, and λ is a parameter that balances the importance between the confidences returned by the ConvNet and the IoU. Thus, s_t^{ij} estimates the likelihood that the i-th detection at frame $t-1$ and the j-th detection at frame t are both true positive detections and come from the same object.

Next, the Viterbi algorithm is applied to obtain the most probable sequences of detections. This algorithm maximizes the conditional probability of the tubelets —each one represents an object seen at different time instants— given a set of detections $\mathscr{D} = \{\mathscr{D}_1, \dots, \mathscr{D}_\tau\}$ and their corresponding scores $\mathscr{S} = \{\mathscr{S}_2, \dots, \mathscr{S}_\tau\}$ over time of the same category. Given the whole set of possible tubelets $\mathscr{V}$, the tubelet with the highest likelihood is:

$$\hat{v} = \arg\max_{v \in \mathscr{V}} \sum_{t=2}^{\tau} s_t^{i(v)j(v)} \tag{3.2}$$

where $i(v)$ and $j(v)$ are the detections at times $t-1$ and t for a given tubelet $v \in \mathscr{V}$.

Once the optimal tubelet $\hat{v}$ is found, those detections within $\hat{v}$ are removed from $\mathscr{D}$ and $\mathscr{S}$, and the process (Equation 3.2) is repeated iteratively to obtain the set of optimal tubelets $\hat{\mathscr{V}}$. Finally, the new confidences for the detections of the last frame τ within each tubelet $\hat{v}$ are updated as:

$$p_{\tau}^{i(\hat{v})} = \frac{1}{\tau}\sum_{t=1}^{\tau} p_t^{i(\hat{v})} \tag{3.3}$$

where $p_t^{i(\hat{v})}$ is the confidence of the detection at time t belonging to tubelet $\hat{v}$. Thus, the confidence of the detections at the last frame are updated with the average confidences of their corresponding tubelets. In this way, tubelet linking increases the confidences of those detections with a low confidence in the last frame, but with a strong track record in previous time instants, which indicates that the detection is a true positive. This process also helps to reduce the confidence of the detections that belong to a tubelet with a weak track record, as this is often supposed to be a false positive. The tubelet linking algorithm is repeated for each category of the dataset.

Correlation-based tubelet linking

Associating the detections in two consecutive frames through IoU might work fine in some scenarios but, in general, it is a very weak feature for object linking. Some scenarios where IoU might generate wrong associations are: small objects that barely overlap between consecutive frames, fast motions of the object and/or the camera, many objects with partial overlaps among them, and videos with a low frame rate or skipping frames.

The proposed correlation-based tubelet linking addresses the preceding points by introducing the correlation score as the feature for data association. In this way, STDnet-ST can associate small objects regardless of their mutual distance in consecutive frames. Also, it is possible to avoid the association of objects with very different features, but placed in the same position in consecutive frames.

Correlation-based tubelet linking modifies Equation 3.1 by replacing the spatial overlap (IoU) with the correlation score to compute the score matrix $\mathscr{S}_t$. Each element of $\mathscr{S}_t$ is calculated as:

$$s_t^{ij} = p_{t-1}^{i} + p_t^{j} + \lambda \cdot c_t^{ij} \tag{3.4}$$

where c_t^{ij} is the correlation obtained by the STDnet-ST ConvNet for the i-th detection at time

$t-1$ and the j-th detection at time t, defined as:

$$c_t^{ij} = \rho(r_{t-1}^{k(i)}, r_t^{l(j)}) \tag{3.5}$$

where ρ represents the correlation module function, and $r_{t-1}^{k(i)}$ and $r_t^{l(j)}$ are the RCN regions at $t-1$ and t associated to detections d_{t-1}^i and d_t^j. So that, $l(j)$ and $k(i)$ are the RoI linking outputs that associate each RCN region r_{t-1}^k and r_t^l with their corresponding detections d_{t-1}^i and d_t^j.

The second novelty is the modification of Equation 3.3 as follows:

$$p_\tau^{i(\hat{v})} = \begin{cases} \max_{t=1}^{\tau} p_t^{i(\hat{v})} & \text{if } \sigma(\{p_t^{i(\hat{v})}\}_{t=1}^{\tau}) \leq \kappa \\ \frac{1}{\tau}\sum_{t=1}^{\tau} p_t^{i(\hat{v})} & \text{otherwise} \end{cases} \tag{3.6}$$

where σ is the standard deviation of the confidences of the tubelet $\hat{v}$, and κ is a threshold. Our hypothesis is that when the confidence variability in a tubelet is small, the last detection might be a true positive and the confidence of that detection can be updated to the maximum confidence of the tubelet. On the other hand, when the variability is high, the confidence is updated with the average confidence, like in the baseline tubelet linking, as the likelihood of being a true positive is lower.

Tubelet suppression procedure

The main downside of the original Viterbi algorithm is that it generates all possible tubelets $\hat{\mathcal{V}}$, even though they are unlikely given their scores. A typical example is a tubelet created with false and true positive detections, only because there is no other possible data association. This causes a decrease in the global accuracy, as discussed in Section 3.4. STDnet-ST tubelet linking manages this situation by defining a tubelet suppression algorithm based on adding dummy detection nodes. Thus, the Viterbi algorithm might build a tubelet using one or more dummy nodes, and these tubelets will be later deleted.

These dummy nodes can be generated owing to the two-level detection —i.e., RCN regions and final detections—, which provides a higher level of abstraction from the RCN regions that do not generate a final detection, but whose correlation score is useful. The tubelet suppression algorithm generates dummy nodes so that: (i) false positives at t are associated to a dummy node rather than to a true positive at $t-1$ or, (ii) true positives at t are associated to a dummy node rather than to a false positive at $t-1$. The first case happens when the dummy

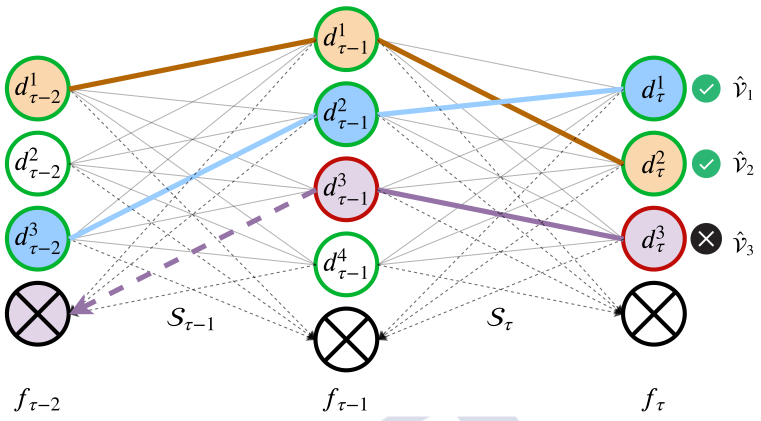


Figure 3.3: An example of the optimal solution provided by the Viterbi algorithm with the tubelet suppression procedure. Nodes with a green border correspond to true positive detections, and those in red with false positive detections. The solution produces a valid tubelet for d^1_τ and d^2_τ (orange and blue) and a non-valid tubelet for d^3_τ (purple).

node has a high correlation with a false positive, e.g., both RCN regions have a similar background. The second case happens when the dummy node has a high correlation with a true positive, e.g., when the RCN region includes an object that was not finally detected. Hence, the gain in the first case is given by the fact that the false positive at t is not associated to true positives and, thus the false positive confidence is not increased. The gain in the second case is given by the fact that the true positive at t is not associated to false positives that decrease its confidence.

Figure 3.3 shows an example of how the Viterbi algorithm works with tubelet suppression. Each node represents a detection d^i_t or a dummy node ($\otimes$). Detections of the same frame are in the same column. Those nodes at different time instants filled with the same color represent the generated tubelets. Solid lines represent the correlation scores (c^{ij}_t) between pairs of detections (Equation 3.5), and dashed lines represent conections between detections and dummy nodes. The tubelet suppression procedure will remove the optimal tubelet $\hat{\mathcal{V}}_3$ as it links the false positive $d^3_{\tau-1}$ with the dummy node in $f_{\tau-2}$ due to the existence of an RCN region $r^i_{\tau-2}$ with a higher correlation score than any other detection in $\tau-2$. Ideally, this indicates that there is a false positive that is detected by the ConvNet at some frames ($\tau-1$ and τ), and filtered out in others ($\tau-2$). If the tubelet linking process does not consider tubelet suppression, the Viterbi algorithm would generate a tubelet including $d^2_{\tau-2}$, $d^3_{\tau-1}$ and

d_τ^3 and, therefore, would probably increase the confidence of d_τ^3, which is a false positive.

Algorithm 1 shows the STDnet-ST tubelet linking algorithm, including the correlation-based tubelet linking and the tubelet suppression procedure. Given the set of detections ($\mathscr{D}$) from time $t = 1$ to $t = \tau$, their confidences ($\mathscr{P}$), and the set of score matrices ($\mathscr{S}$) —s_t^{ij} (Equation 3.4) is the ij element of matrix $\mathscr{S}_t$, calculated from the i-th detection at time $t-1$ and the j-th detection at time t—, the algorithm returns the updated confidences ($\hat{\mathscr{P}}_\tau$) associated to the detections at time τ. First, we initialize $\hat{\mathscr{P}}_\tau$ with the confidences generated by the ConvNet (line 1). Then, we add a dummy node (line 3) to the detection set at time t —with original size n_t— as well as one column (line 5) and one row (line 6) to the score matrix at time t —with original size $n_{t-1} \times n_t$. In the added column we store the scores between a dummy node and all detections at $t-1$, while in the added row are the scores between a dummy node and the detections at t.

The scores associated with dummy nodes are based on Equation 4 and Equation 5, but where one of the two RCN regions involved in Equation 5 is a free RCN region —RCN region without detection— (lines 8 and 11). The free RCN regions are those that have been discarded by the ConvNet because the likelihood of containing an object is low. In particular, for each of the detections at $t-1$, the free RCN region from t that will be selected to compute the correlation score is the one with the maximum correlation score. The same for the detections at t and the free RCN regions from $t-1$. So that, the correlation (line 8) for a given detection d_{t-1}^i within an RCN region $r_{t-1}^{j(i)}$ is the maximum correlation between $r_{t-1}^{j(i)}$ and the whole set of free RCN regions at t (r_t^k). Then, new scores (lines 9 and 12) are calculated as in Equation 4, where p_{t-1}^i and p_t^j both come from the real detection d_{t-1}^i, i.e, $p_{t-1}^i = p_t^j$.

Next, the Viterbi algorithm is applied with the set of detections and the new set of score matrices (line 14), while every $\mathscr{D}_t$, from $t = 1$ to $t = \tau$, still has detections provided by the STDnet-ST ConvNet —not just dummy nodes— (line 13). Then, for each generated tubelet by the Viterbi algorithm, the corresponding detections are deleted from the set of detections (line 18), and the corresponding row and column is also deleted from the score matrices (lines 20 and 22). A detection at time t contributes to the score matrices $\mathscr{S}_t$ and $\mathscr{S}_{t+1}$ —Figure 3.3. Finally, if the tubelet is valid, i.e., it does not contain dummy nodes, the confidences of the detections at time τ that are in the set of tubelets are updated following Equation 3.6 (line 26).

Algorithm 1: STDnet-ST tubelet linking

```
   Input  : $\mathscr{D} = \{\mathscr{D}_t = \{d_t^1, \dots, d_t^{n_t}\} \mid t = 1, \dots, \tau\}$
   Input  : $\mathscr{P} = \{\mathscr{P}_t = \{p_t^1, \dots, p_t^{n_t}\} \mid t = 1, \dots, \tau\}$
   Input  : $\mathscr{S} = \{\mathscr{S}_t = \{s_t^{11}, \dots, s_t^{n_{t-1}n_t}\} \mid t = 1, \dots, \tau\}$
   Output: $\hat{\mathscr{P}}_\tau$
1  $\hat{\mathscr{P}}_\tau \leftarrow \mathscr{P}_\tau$
2  for $t = 1, \dots, \tau$ do
3      $\mathscr{D}_t \leftarrow \mathscr{D}_t \cup d_t^{\varnothing}$
4      if $t > 1$ then
5          $\mathscr{S}_t \leftarrow \mathscr{S}_t : d_{t-1}^{\varnothing\star}$
6          $\mathscr{S}_t \leftarrow \mathscr{S}_t : d_t^{\star\varnothing}$
7          for $i = 1, \dots, n_{t-1}$ do
8              $c_t^{i,n_t+1} = \max_k \rho(r_{t-1}^{j(i)}, r_t^k) \mid r_t^k \not\rightarrow d_t^l \; \forall \, l = 1, \dots, n_t$
9              $s_t^{i,n_t+1} = p_{t-1}^i + p_{t-1}^i + \lambda \cdot c_t^{i,n_t+1}$
10         for $i = 1, \dots, n_t$ do
11             $c_t^{n_{t-1}+1,i} = \max_k \rho(r_{t-1}^k, r_t^{j(i)}) \mid r_{t-1}^k \not\rightarrow d_{t-1}^l \; \forall \, l = 1, \dots, n_{t-1}$
12             $s_t^{n_{t-1}+1,i} = p_t^i + p_t^i + \lambda \cdot c_t^{n_t+1,i}$
13 while $\{d_t^1 \neq d_t^{\varnothing} \; \forall \, t = 1, \dots, \tau\}$ do
14     $\hat{v} \leftarrow$ Viterbi($\mathscr{D}, \mathscr{S}$)
15     isvalid $\leftarrow$ True
16     for $t = 1, \dots, \tau$ do
17         if $d_t^{i(\hat{v})} \neq d_t^{\varnothing}$ then
18             $\mathscr{D}_t \leftarrow \mathscr{D}_t \setminus d_t^{i(\hat{v})}$
19             if $t > 1$ then
20                 $\mathscr{S}_t \leftarrow$ deleteColumn($\mathscr{S}_t, i(\hat{v})$)
21             if $t < \tau$ then
22                 $\mathscr{S}_{t+1} \leftarrow$ deleteRow($\mathscr{S}_{t+1}, i(\hat{v})$)
23         else
24             isvalid $\leftarrow$ False
25     if isvalid then
26         $\hat{p}_\tau^{i(\hat{v})}$ = updateConfidence($\mathscr{P}, \hat{v}$) [Equation 3.6]
27 done
```

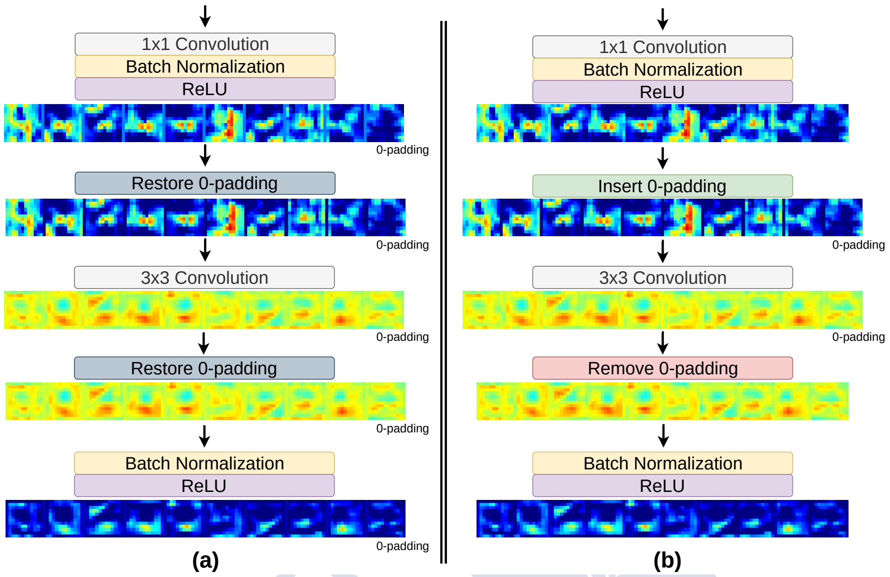


Figure 3.4: Structure of a ResNet block for STDnet: (a) using the original 0-padding implementation and (b) with the 0-padding implementation proposed in this chapter.

3.3.3 Spatial STDnet enhancement

This section describes the improvements made over the original version of STDnet [9]: a new 0-padding operation between regions, and the replacement of the classical header with a cascaded header.

Rethinking the 0-padding operation

The first improvement concerns the structure of the convolution blocks after obtaining the promising regions by the Region Context Network (RCN) and the RoI Collection Layer (RCL). In the original STDnet, the RCL encompasses the different regions proposed by the RCN and adds a 0-padding between them so that the convolution kernels larger than 1×1 do not share information from adjacent regions.

Figure 3.4(a) shows how the 0-padding was restored before and after each convolution. However, although the convolution operations were not affected with this structure, 0-padding restoration did have an effect on batch normalization and harmed training. For instance, if there are 50 RCN regions with a size of 8×8 and 1px 0-padding between every pair, the overall

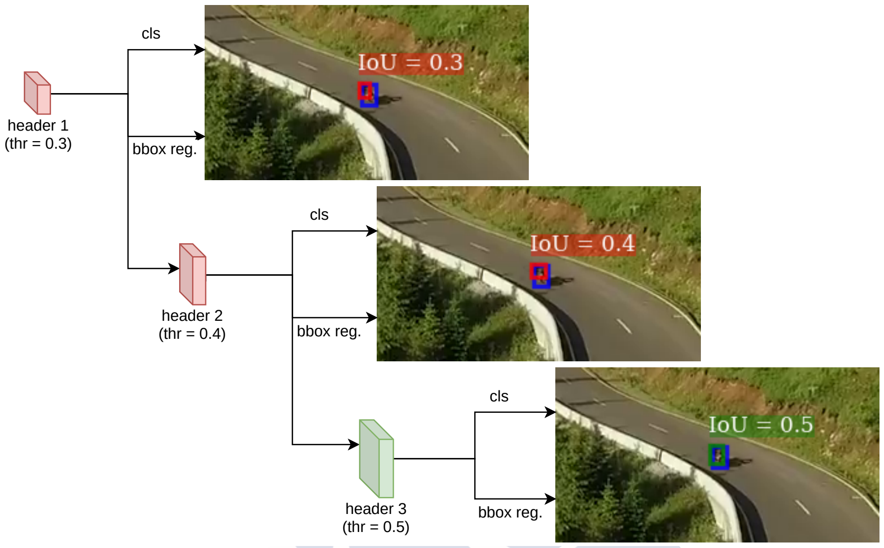


Figure 3.5: Performance of the cascaded header over a rough proposal towards a true positive detection.

size of each channel will be 449×8 = 3592px[2], where 49×8 = 392px are 0's. Thus, all those 0's —which are more than the 10% of the pixels in the feature map— were influencing the learning of the network. In addition, 1×1 convolution operations had to perform unnecessary operations on that 0-padding.

To solve this problem, a built-in operator has been implemented that inserts and removes the 0-padding before and after each convolution greater than 1×1. The new structure is represented in Figure 3.4(b).

Cascaded header

The original STDnet header [9] (classifier + bounding box regression) has been replaced by three consecutive headers that interatively improve small objects detection. This implementation is based on the research carried out by [13], where several twin headers are trained with progressively more restrictive overlap thresholds.

[2]449×8 is the shape of a feature map channel composed of 50 regions arranged horizontally with 1px padding between each pair: width = $(50\times8) + (1\times49)$; height = 8;

The idea is to improve in successive headers object detections that have a minimum overlap with the true object until reaching the threshold defined to be considered true positive. Therefore, this will improve not only the final accuracy, but also the final recall (Figure 3.5). A common problem in small object detection are double detections: for large objects, non maximum suppression eliminates those double detections but, for small objects, the overlap is very small. The cascaded header will help to eliminate these false positives, as bounding boxes will be more accurate.

Differently from the implementation in [13], where the cascade is applied directly on the feature map prior to RPN, the additional headers that make up the cascade approach in STDnet-ST take the information from the same feature map, but with disjoint regions. The cascaded headers have to compute the target bounding box using the predecessor header, and have to retrieve the spatial information of the regions relative to the input image from the Region Context Network (RCN).

3.4 Experiments

3.4.1 Evaluation metrics

We assess the performance of our approach and previous work with the metrics reported in MS COCO [80]. Such metrics are the Average Precision ($AP^{@.5}$), which gives the average precision of those objects detected with at least 50% IoU between the detected and the ground-truth bounding boxes, and $AP^{@[.5,.95]}$, which is the average AP when the IoU goes from 50% to 95% in 5% steps. In the default COCO metrics, the results are shown for three different subsets: *small* (AP_s), objects smaller than 1,024 pixels area; *medium* (AP_m), objects between 1,024 and 9,216 pixels area, and *large* (AP_l), objects larger than 9,216 pixels area. In this chapter we define a new scale subset following COCO style, *extremely small* (AP_{xs}), to include small targets as defined in this chapter, i.e., those enclosed in bounding boxes with less or equal than 256 pixels area. The *XS* subset is defined in order to evaluate the performance for extremely small objects.

3.4.2 Datasets

We conduct extensive experiments on three publicly accessible datasets: USC-GRAD-STDdb [9], UAVDT [27] and VisDrone2019-VID [153].

- **USC-GRAD-STDdb** [9]. It comprises 115 video segments with more than 25,000 annotated frames. The resolution of the video is HD 720p (1,280 $\times$ 720). There are more than 56,000 objects, with most of them ranging from 16 ($\approx 4 \times 4$) up to 256 ($\approx 16 \times 16$) as pixel area, i.e., small objects as defined in this chapter. The videos in USC-GRAD-STDdb comprise three main landscapes —air, sea and land— with five object categories, namely: air (drone, bird), 57 videos with 12,139 frames; sea (boat), 28 videos with 7,099 frames; and land (vehicle, person), 30 videos with 6,619 frames. Nevertheless, the evaluation will be carried out as a single category. The test subset holds 11,337 objects, where almost 90% of them (10,136 objects) correspond to the *extremely small* subset.

- **UAVDT** [27]. It contains 23,829 frames of training data and 16,580 images of test data of $\approx$ 1,024 $\times$ 540 resolution. The videos are recorded with an UAV platform over different urban areas. The ground truth targets are vehicles labeled as car, bus and truck, but evaluated as a single category. UAVDT comprises a total of 375,884 test objects, where 76,215 are considered within the *extremely small* subset (20.3%).

- **VisDrone2019-VID** [153]. The VisDrone2019-VID challenge provides a total of 96 HD/Full HD video sequences, including 56 sequences for training (24,201 frames in total), 7 sequences for validation (2,819 frames in total) and 17 sequences for development testing (test-dev) (6,635 frames in total). There is also a blind test (test-challenge) subset that comprises 16 videos, but the evaluation system does not report the metrics for the *extra small* subset, so these data will be dismissed and the results will be reported only for the test-dev subset. The dataset is mainly focused on people and vehicles, where ten diffent categories of interest are labeled: pedestrian, person, car, van, bus, truck, motor, bicycle, awning-tricycle, and tricycle. The 17 sequences for testing hold 310,228 test objects, where 27,027 (8.7%) are considered *extra small*.

3.4.3 Implementation Details

We implemented STDnet-ST based on STDnet [9]. Faster R-CNN [106] with Feature Pyramid Network (FPN) [78] is adopted as the baseline detection network for small object detection. We have also compared our proposal with the state-of-the-art spatio-temporal approaches: FGFA [154], RDN [23] and MEGA [16][3], with the anchors' size best suited for small objects

[3]https://github.com/Scalsol/mega.pytorch

as defined in [9]. These three approaches achieve state-of-the-art results in ImageNet VID dataset [110].

Training phase. The input size of STDnet-ST is determined by the resolution of the video under study, namely, 1,280 $\times$ 720 pixels in USC-GRAD-STDdb, 1,024 $\times$ 540 in UAVDT and 1,920 $\times$ 1,080 in VisDrone2019-VID. For USC-GRAD-STDdb, as most of the objects belong to the *XS* size, i.e., below 256 pixels area, RCN regions of size 32 $\times$ 32 suffice to enclose all objects. In these cases the STDnet-ST training phase is continuous during 40k iterations with two step decay. For UAVDT and VisDrone2019-VID, with objects with more varying sizes, including those larger than the *XS* category, i.e., below 256 pixels area, the training process requires pre-training. Thus, first, we run a pre-training phase with Faster R-CNN during 20k iterations to address all object sizes followed by a fine-tuning with STDnet-ST for other 20k iterations with two step decay. In order to retrieve all objects with more diverse aspect ratios, we set the RCN region size to 48 $\times$ 48 pixels. Also, as reported in [9], for both datasets, RCN between *conv3* and *conv4* and the initialization of anchors by k-means lead to the best performance metrics. Finally, when training the model, we set the base learning rate to 0.0025, a momentum of 0.9, and parameter decay of 0.0001 on weights and biases.

Test phase. The input size and the RCN region size are the same as those of the training phase. The maximum number of RCN regions is set to 100. The spatio-temporal hyperparameters τ, κ and λ are set to 4, 0.02 and 1.0, respectively, derived by experimental studies over a validation subset from the USC-GRAD-STDdb training set. We also apply a box-voting scheme after non-maximum suppression [38].

In addition, there are some differences between the original STDnet [9] and the STDnet used as base of STDnet-ST for this chapter: (1) STDnet-ST has been implemented in Caffe2; (2) the RoI pooling dimension is reduced from 7$\times$7 to 4$\times$4 when the cascaded header is applied to keep constant the computational cost; (3) the last ResNet convolutional block (*conv5*) is replicated to fine-tune each header during the training phase; and (4) all possible positive RCN regions are passed through the network instead of limiting their number, in order to perform well in datasets with many objects of interest per image.

3.4.4 Results on USC-GRAD-STDdb

Table 3.1 and Table 3.2 show experimental results on USC-GRAD-STDdb [9]. Our approach is compared to the state-of-the-art FPN [78], as it proved to be the most competitive method for the present dataset [9], and FPN with cascaded header (Cascade-FPN [13]), for fair com-

Table 3.1: Ablation study on USC-GRAD-STDdb for the different tubelet linking components of STDnet-ST++. Results without correlation features are implemented directly over one branch of STDnet-ST++ —i.e., the first three rows—, and those that use them represent the different versions of STDnet-ST++ —i.e., the last four rows. The first row refers to STDnet as it is defined in [9] with the enhancements proposed in Section 3.3.3, i.e., STDnet++.

Baseline linking	Confidence variability	Correlation linking	Tubelet suppression	$\text{AP}_{xs}^{@[.5,.95]}$	$\text{AP}_{xs}^{@.5}$
—				18.9	59.1
✓				20.1	61.4
✓	✓			20.3	61.8
		✓		20.4	61.6
	✓	✓		20.6	62.0
		✓	✓	20.9	62.6
	✓	✓	✓	**21.4**	**63.4**

parisons. From here on, the names of the different versions of STDnet are as follows: STDnet refers to the original ConvNet; STDnet++, refers to STDnet with the enhancements detailed in Section 3.3.3; STDnet-ST and STDnet-ST++ refer to the spatio-temporal architectures defined in Section 3.3.1 and Section 3.3.2 adopting STDnet and STDnet++ as base network, respectively. Finally, as the baseline tubelet linking and the confidence variability methods are independent of the architecture, we have also tested the performance of FPN and Cascade-FPN with these components —referred as FPN-t and Cascade-FPN-t.

Table 3.1 studies the influence of the different components defined in this chapter to exploit the temporal information from a video dataset. *Baseline linking* refers to the baseline method to generate tubelets defined in Section 3.3.2; *Confidence variability* refers to the modification of the confidences of the detections based on the confidences of the tubelets due to their variability, as addressed in Equation 3.6; *Correlation linking* means the correlation-based tubelet linking as addressed in Section 3.3.2; and *Tubelet suppression* concerns the tubelet suppression procedure presented in Section 3.3.2.

As it can be observed, the use of temporal information leads to higher performance. STDnet-ST++ outperforms STDnet++ from 18.9% to 21.4% for $\text{AP}_{xs}^{@[.5,.95]}$ and from 59.1% to 63.4% for $\text{AP}_{xs}^{@.5}$. In this ablation study, it is also possible to determine the contribution of each of the components to the performance of STDnet. The correlation-based linking, together with the confidence variability contribute to increase 0.5% $\text{AP}_{xs}^{@[.5,.95]}$ and 0.6% $\text{AP}_{xs}^{@.5}$

Table 3.2: Evaluation metrics for different methods on USC-GRAD-STDdb database. -t indicates the use of baseline tubelet linking and confidence variability to compute the final score of each tubelet.

Method	$AP_{xs}^{@[.5,.95]}$	$AP_{xs}^{@.5}$
FGFA [154]	5.3	20.4
RDN [23]	14.1	46.3
MEGA [16]	15.1	46.8
FPN [78]	17.3	54.5
Cascade-FPN [13]	17.4	55.9
FPN-t	18.7	57.2
Cascade-FPN-t	19.1	58.9
STDnet [9]	18.3	57.8
STDnet++	18.9	59.1
STDnet-ST	20.1	62.1
STDnet-ST++	**21.4**	**63.4**

—Table 3.1, rows 2 and 5. Also, the tubelet suppression procedure adds a gain of 0.8% $AP_{xs}^{@[.5,.95]}$ and 1.4% $AP_{xs}^{@.5}$ over the previous result —Table 3.1, rows 5 and 7.

Two conclusions can be drawn from Table 3.1. First, the importance of the correlation obtained by the ConvNet of STDnet-ST. The correlation-based tubelet linking is capable of improving the IoU-based baseline tubelet linking by comparing the early features of the objects and their context; and more importantly, it allows to build the tubelet suppression procedure with a higher level of abstraction that cannot be found by associating detections by spatial overlap. Second, the importance of the confidence variability when combined with the tubelet suppression procedure, as some of the tubelets that were composed by false negatives are discarded and, therefore, the confidence variability is more reliable.

Table 3.2 provides a comparison in terms of accuracy between the state-of-the-art FGFA, RDN, MEGA, FPN, Cascade-FPN, and our architectures STDnet-ST and STDnet-ST++. STDnet-ST++ outperforms FPN by 4.1% $AP_{xs}^{@[.5,.95]}$ and 8.9% $AP_{xs}^{@.5}$ and Cascade-FPN by 4.0% $AP_{xs}^{@[.5,.95]}$ and 7.5% $AP_{xs}^{@.5}$. FPN with spatio-temporal information improves its baseline by 1.4% $AP_{xs}^{@[.5,.95]}$ and 2.7% $AP_{xs}^{@.5}$. Even so, the results of the spatio-temporal FPN and Cascade-FPN remain below STDnet-ST and STDnet-ST++. When compared to the spatio-temporal approaches FGFA, RDN and MEGA, STDnet-ST++ outperforms them by at least 6.3% $AP_{xs}^{@[.5,.95]}$ and 16.6% $AP_{xs}^{@.5}$. This is mainly due to the fact that both the RPN placed in

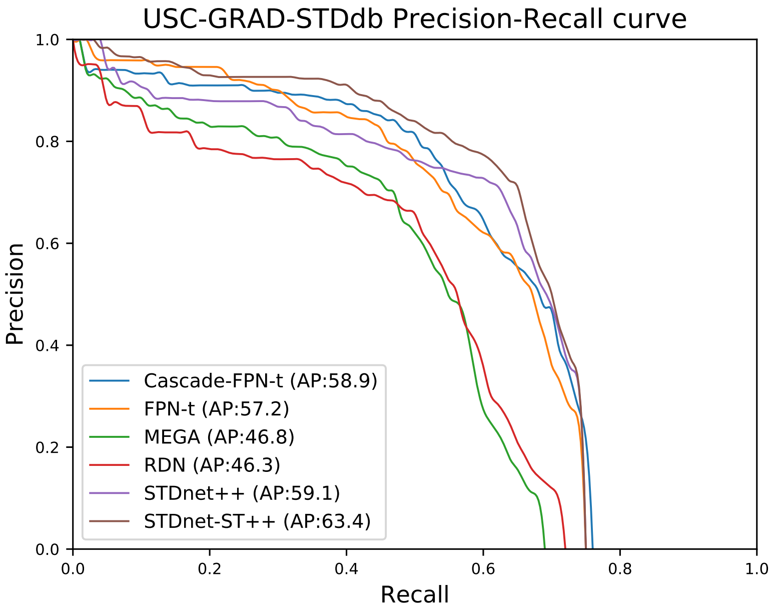


Figure 3.6: Precision-Recall curves and $AP_{xs}^{@.5}$ of the most relevant approaches in Table 3.2 for USC-GRAD-STDdb.

deep layers and the association methods used in these approaches have a high performance on large objects but a lower impact when dealing with extremely small objects. With respect to the original version of STDnet, STDnet-ST++ improves the result by a 3.1% $AP_{xs}^{@[.5,.95]}$ and a 5.6% $AP_{xs}^{@.5}$. The most relevant results are shown in Figure 3.6 using Precision-Recall curves.

3.4.5 Results on UAVDT

The experimental results on the UAVDT dataset [27] are shown in Table 3.3. The first four rows are computed using the bounding box results provided in [27], and directly adapted to the MS COCO results format [80].

Results confirm that the spatial STDnet performs better than the rest of the state-of-the-art spatial approaches. Moreover, it can be seen how the enhancements introduced for STDnet (STDnet++) improve $AP_{xs}^{@[.5,.95]}$ by 0.6% and $AP_{xs}^{@.5}$ by 2.9% higher than any other spatial approach. $AP^{@[.5,.95]}$ is considered the primary challenge metric by MS COCO [80], because it encompasses AP adding information on how it behaves as the IoU reaches perfection. It

Table 3.3: Evaluation metrics on the *extremely small* subset of UAVDT dataset, i.e., objects under 16 × 16 pixels.

Method	$AP_{xs}^{@[.5,.95]}$	$AP_{xs}^{@.5}$
Faster R-CNN [27]	6.6	26.0
R-FCN [27]	9.2	32.5
RON [27]	3.7	19.7
SSD [27]	6.0	23.5
FGFA [154]	5.8	19.0
RDN [23]	9.0	27.6
MEGA [16]	9.8	28.8
FPN [78]	11.8	29.7
FPN-t	12.0	30.3
Cascade-FPN [13]	12.0	30.5
Cascade-FPN-t	12.3	31.2
STDnet [9]	12.5	35.1
STDnet++	12.6	35.4
STDnet-ST	13.1	36.0
STDnet-ST++	**13.3**	**36.4**

is also noteworthy that STDnet outperforms FPN-t and Cascade-FPN-t, which exploit spatio-temporal information.

As expected, our spatio-temporal proposal, STDnet-ST++, accomplishes better performance with respect to its spatial version, achieving state-of-the-art results in the UAVDT dataset for the *extremely small* subset. STDnet-ST++ overcomes spatio-temporal Cascade-FPN (Cascade-FPN-t) by 1.0% $AP_{xs}^{@[.5,.95]}$ and 5.2% $AP_{xs}^{@.5}$, and also R-FCN by 4.1% $AP_{xs}^{@[.5,.95]}$ and 3.9% $AP_{xs}^{@.5}$. Figure 3.7 shows the Precision-Recall curves.

3.4.6 Results on VisDrone2019-VID

The experimental results on the Visdrone2019-VID dataset [153] are shown in Table 3.4. In first place, it is confirmed that the STDnet based approaches outperform their counterparts. In second place, STDnet++ improves STDnet by 0.1% $AP_{xs}^{@[.5,.95]}$ and by 0.6% $AP_{xs}^{@.5}$. Finally, regarding the spatio-temporal approaches, STDnet-ST++ boosts 0.2% $AP_{xs}^{@[.5,.95]}$ and 0.4% $AP_{xs}^{@.5}$ its baseline, while improving 1.2% $AP_{xs}^{@[.5,.95]}$ and 2.0% $AP_{xs}^{@.5}$ compared to the best FPN-based approach. Examples of detections with STDnet-ST++ on the three datasets

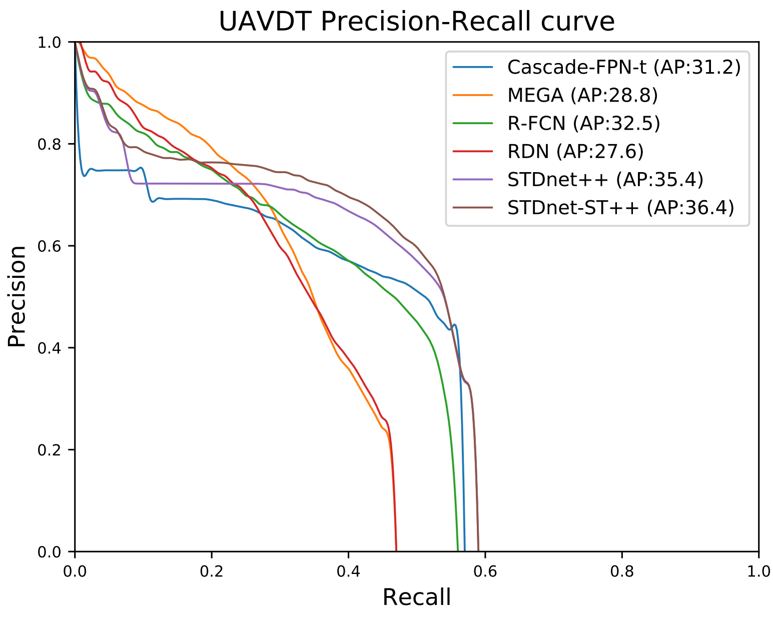


Figure 3.7: Precision-Recall curves and $AP_{xs}^{@.5}$ of the most relevant approaches in Table 3.3 for UAVDT.

reported in this chapter are shown in Figure 3.8.

3.5 Conclusion and future work

We have introduced STDnet-ST, a spatio-temporal ConvNet to detect small targets in video. STDnet-ST is composed of two branches, and it binds the detections of two input frames by a correlation module to create spatio-temporal small object tubelets. Those tubelets are refined at the tubelet linking stage, which applies the Viterbi algorithm to the detections based on correlation linking, and implements a tubelet suppression procedure that allows STDnet-ST to dismiss unprofitable tubelets while preserving only high quality ones.

Furthermore, certain components of the STDnet structure [9] have been reformulated, leading to the definition of STDnet++ and STDnet-ST++. Enhancements have been made to 0-padding operation, for improving the network learning, and a cascaded header, to obtain better performance by turning false positives with low overlap into true positives.

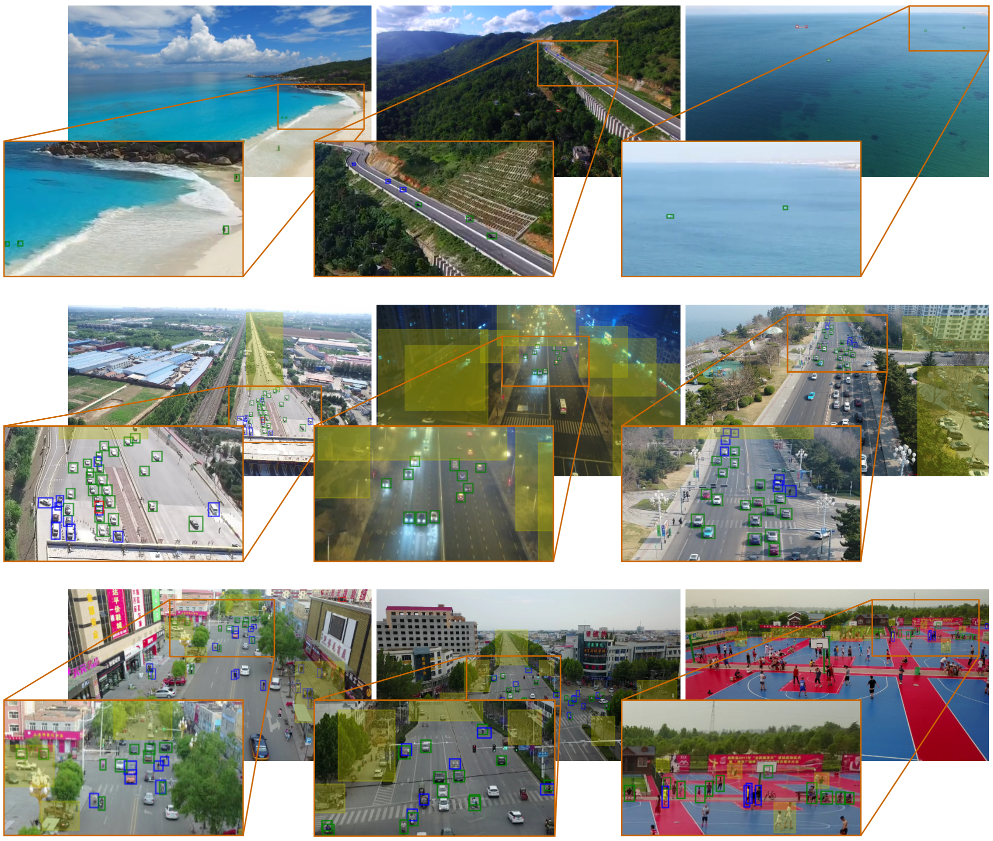


Figure 3.8: Some object detection results of STDnet-ST++ for USC-GRAD-STDdb (top), UAVDT (middle) and VisDrone2019-VID (bottom) test sets. A confidence threshold of 0.6 was used to display these images. For each image, green boxes are true positives, red boxes false positives and blue boxes false negatives. The yellow rectangles are ignored regions. Only objects that belong to the *XS* size are displayed.

Table 3.4: Evaluation metrics on the *extremely small* subset of VisDrone2019-VID dataset, i.e., objects under 16 $\times$ 16 pixels.

Method	$AP_{xs}^{@[.5,.95]}$	$AP_{xs}^{@.5}$
FGFA [154]	2.5	11.0
RDN [23]	3.8	17.1
MEGA [16]	4.0	17.2
FPN [78]	6.2	19.9
Cascade-FPN [13]	6.1	20.2
FPN-t	6.3	20.2
Cascade-FPN-t	6.2	20.4
STDnet [9]	7.2	21.4
STDnet++	7.3	22.0
STDnet-ST	7.5	21.9
STDnet-ST++	**7.5**	**22.4**

In order to validate the proposed architecture, we have conducted experiments over three publicly available datasets with a large number of small objects: USC-GRAD-STDdb [9], UAVDT [27] and VisDrone2019-VID [153]. STDnet-ST++ achieves state-of-the-art results in all these datasets for extremely small objects, clearly outperforming its counterparts by 2.3% $AP_{xs}^{@[.5,.95]}$ on USC-GRAD-STDdb, by 1.0% $AP_{xs}^{@[.5,.95]}$ on UAVDT and by 1.2% $AP_{xs}^{@[.5,.95]}$ on VisDrone2019-VID.

Results show how the three main characteristics of STDnet-ST are key to achieve small object detection: (i) the use of high resolution feature maps throughout the architecture allows to locate the objects and adjust their bounding boxes; (ii) performing correlation over RCN regions allows to correctly associate objects in two consecutive frames, therefore, improving the detection precision; (iii) the correlation-based tubelet linking together with tubelet suppression procedure provide high quality tubelets to increase the final accuracy. The tubelet suppression procedure is possible due to the RCN regions, that provide a limited number of areas without objects where to look for possible correlations with false positive detections, therefore avoiding their linking with true positive detections.

As future work, we plan to address the limited number of small objects present in current datasets. Considering that manual object annotation is extremely time-consuming and that tracking-based annotation is far from being perfect, we will work on the definition of a pipeline to generate synthetic small objects from larger ones. Specifically, the recent advances in Generative Adversarial Networks (GANs) seem to be a promising route both for the generation of synthetic objects close to real ones and for suitable placement in different contexts. Super-resolution GANs are attractive in the former task, and inpainting and blending GANs for the latter one.

CHAPTER 4

DATA AUGMENTATION FOR SMALL OBJECT DETECTION THROUGH A DOWNSAMPLING GAN

In Chapters 2 and 3 we introduced the approach for small object detection in images and videos. In this chapter we explore a different way to improve the detection accuracy for small objects: adding synthetic examples that boost the precision. This approach is even more useful when the number of examples is reduced, as it happens for small objects under 32×32 pixels in many datasets. Although we have released the Small Target Detection database (USC-GRAD-STDdb), the effort involved in annotating a large amount of data makes it unfeasible as a long-term methodology. The advent of the generative adversarial networks (GANs) opens up a new data augmentation possibility for training architectures without the costly task of annotating huge datasets for small objects.

In this chapter, we propose a full pipeline for data augmentation for small object detection which combines techniques of adversarial training, object segmentation, image inpainting and image blending to achieve high quality synthetic data. The main component of our pipeline is DS-GAN, a novel GAN-based architecture that generates realistic small objects from larger ones. In addition to the synthetic data generation, the pipeline combines object segmentation, object inpainting and object blending to find a proper place into the new image —a place not overlapped with other objects and with a suitable context—, and to adapt the object to the new background —i.e., without abrupt edges and with color consistencies.

The contents of this chapter are extracted from the following publication:

B. Bosquet, L. Seidenari, V. M. Brea, M. Mucientes, and A. Del Bimbo. Data augmentation for small object detection through a downsampling GAN. Article submitted to IEEE Transactions on Image Processing in October 2020 and currently under review.

4.1 Introduction

Object detection is a fundamental technique within computer vision, as identifying objects in images or videos is mandatory for image understanding and it is used in many real word applications, including self-driving cars, unmanned aerial vehicles (UAVs), satellite image analysis, or anomaly detection in medical images. The accuracy of detectors has experienced a lot of progress year on year since the release of large training datasets and, more importantly, since the continuous improvement of convolutional neural netwoks (CNNs) architectures [39, 40, 47, 46, 106], which goes along with the ever increasing computing power of Graphic Processing Units (GPUs).

In this line, small object detection stands out as a field of its own with increasing interest [2, 8, 69]. This is mainly because many downstream tasks demand early detections of objects to act quickly: self-driving cars or applications like sense and avoid on UAVs need to detect as far an object as possible or satellite image analysis, where almost all objects are just a few pixels in size. That is, all the previous applications require objects to be identified as soon as possible, i.e, when they are barely visible in the images. Recent CNN-based object detectors, like the work in [78], provide high accuracy over a wide range of scales, from less than 32×32 pixels up to the image size. Despite these improvements, existing solutions often underperform with small objects [82, 90].

The problems of detecting such small objects are twofold: (i) in deep CNNs architectures commonly the deeper the feature map, the lower the resolution, which is counterproductive when the object is so small that it may be lost along the way, and (ii) the most popular datasets such as MS COCO [80] or ImageNet [110] focus their attention on larger objects. While to deal with the first problem new solutions are being proposed year by year [2, 9, 78, 147], the second is being tackled mostly with the tedious task of generating new datasets [8, 27, 153].

We have noticed some reasons that call for a superior number of small objects in public datasets to train a small object detector. First, the relatively fewer images that contain small

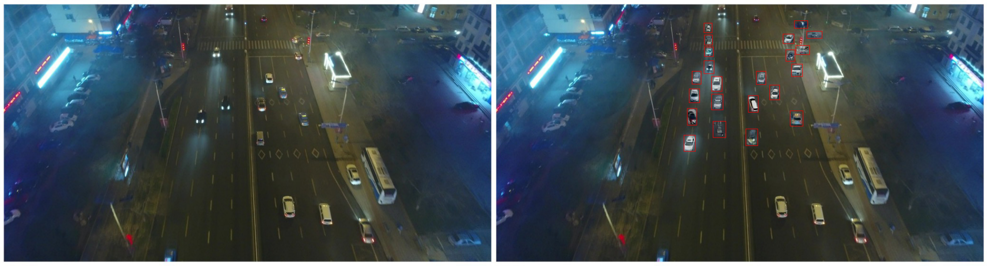


Figure 4.1: We describe a pipeline for small object data augmentation that is able to populate an original frame (left) with new generated small objects (right). New objects are highlighted with red bounding boxes.

objects will potentially bias any detection model to focus more on medium and large objects. In addition, the scarce features in small objects hinder the model generalization, lacking a great deal of variability. Finally, the smaller the object the more places it can appear, increasing the object background diversity and, thus, demanding more context variability at training.

Moreover, pieces of evidence [156] have shown that good data augmentation can boost deep models to achieve state-of-the-art performance without changing the network architecture. Although data augmentation has shown to significantly improve image classification, its potential has not been thoroughly investigated for object detection. So, given the additional cost for annotating images for object detection, data augmentation may play an essential role in boosting performance of generic object detection.

The advent of the generative adversarial networks (GANs) [42] has led to a new approach in the field of data augmentation. This kind of models are trained in an adversarial manner, where one network (the generator) tries to cheat another network (the discriminator) by generating new images. Through an iterative process, the generator will attempt to provide examples that are increasingly similar to those in the real world.

Data augmentation for object detection presents two major challenges: (i) the generation of new objects and (ii) the integration of those objects to adapt them to the new scenarios. The former is mostly tackled by reusing already existing objects in different positions [69] or by adjusting their scale by re-scaling functions [15]. However, it has been proven that common re-scaling functions cause artefacts that significantly distort the re-scaled object if compared to real-world objects [10, 113]. The latter could be approached by object segmentation methods [46] to clear the original background and then insert the objects into plausible positions while tuning for color consistencies. In the case of small objects, there is the added issue that the performance of the segmentation methods decreases dramatically. In addition, many popular datasets [9, 27, 153] do not contain segmentation ground truth to train the segmentation models properly.

For all these reasons, in this chapter we will propose a full pipeline for small object data augmentation. The pipeline takes a video dataset as input and returns the same dataset but with new synthetic small objects (Figure 4.1) —without looking for a temporal consistency between them. The hypothesis is that, starting from the visual features of larger objects —which can be found in many datasets in a large number—, high quality synthetic small objects can be generated and placed into an existing image. To do so, the pipeline has the following stages: (i) generate small objects from large ones through a GAN; (ii) seek a logical position within the image through optical flow; (iii) integrate small object via inpainting and blending techniques. The main contributions described in this chapter are:

- A full pipeline for small object data augmentation which is able to automatically generate small objects using larger ones and place them into an existing background in a congruent fashion.
- DS-GAN, a generative adversarial network architecture that converts large size objects into high quality small objects.
- An extensive experimentation on the public video dataset UAVDT [27], where the base results of state-of-the-art approaches are improved.

4.2 Related Work

The small object data augmentation approach we present in this chapter is based on several computer vision tasks. The execution flow starts with a generative adversarial network (GAN) that generates synthetic small objects from larger ones. This process can be considered as the opposite problem to the well-known image super-resolution techniques. Then, a segmentation network obtains the pixels of the input object and this mask is adapted to the new generated small object. In parallel, the new position in the image is obtained using optical flow techniques. The synthetic object may or may not replace an existing small object in the image. If so, the real one is removed from the scenario by inpainting techniques. Finally, the object is placed into the selected position and tuned by image blending to fit the new background.

Small object detection. Small object detection refers to improving the detection of those objects with small size and poor visual features, typically defined as the detection of objects with a size below 32×32 pixels [80]. Small object detection is closely related to the impressive results in object detection via CNN-based architectures [78, 79], but with particular variations as discussed in Chapters 2 and 3. The actual trend for common object detection is to go deeper to recognise more complex semantics [47], but small objects, that do not contain detailed visual features, may be lost in the deep network. More sophisticated architectures such as the Feature Pyramid Network (FPN) [78] or the Region Context Network (RCN) [8] partially alleviates this problem.

Furthermore, another restriction is the fact that popular datasets have focused their attention on larger objects, and small objects are underrepresented [80, 110]. To some extent, this restriction has been reduced by the advent of video datasets recorded by UAVs with built-in cameras over wide areas in the wild with small objects and decent quality. In particular, UAVDT [27], VisDrone2019-VID [153] and, especially, USC-GRAD-STDdb [8] are video datasets with a large percentage of small objects.

In spite of efforts to design new architectures or the release of more training data, small object detection accuracy lags behind that of larger objects [82], which opens the way for more improvement in both areas.

Data augmentation. Data augmentation strategies are widely used for training vision models to minimize the bias between the training and the testing subsets, i.e., leading to more generalized models. There are two main types of data augmentation: basic image manipulations and generative synthetic approaches. Basic image manipulations encompass those methods that add some distortions to a real image by geometric transformations, color space transformations, noise injection or random erasing. Generative synthetic comprises more sophisticated approaches that aim to learn possible characteristics from the training data to create similar synthetic instances.

Basic manipulations are simple operations, so deep learning designs usually combine many of them. In object recognition the geometric transformation of scale, translation, and rotation [18, 111, 127] as well as cropping, image mirroring and color processing [72] are very standardized. For object detection, image mirroring and object-centric cropping are the most widely used [83, 116]. The random erasing [151] technique aims to improve the robustness of the model to partially occluded samples by randomly adding occlusion, which also reduces the risk of over-fitting.

One straightforward solution to generate synthetic objects is studied by Kisantal *et al.* [69], which aims to augment the number of small object instances by copy-pasting them. They firstly use segmentation masks to crop small objects, and then randomly paste the cropped small objects in the image. The problem of this approach is twofold: (i) the features of the object remain the same and (ii) the position and scale of the object may not fit the context —e.g., a car in the sky. The second issue is addressed in [15] by an adaptive augmentation strategy called AdaResampling that logically augments the instances. AdaResampling generates a prior context map using a segmentation CNN and then places the objects in accordance with the scale and position. Yet, [10, 113] show that the object features produced by conventional resizing functions are far from real-world object features.

Another solution is to learn the space of possible augmentations with adversarial training. Adversarial learning attempts to fool models through malicious input or adversarial attacking through two —or more— networks with contrasting objectives. These samples are usually generated by looking for the minimum possible noise injection needed to cause a misclassification [88]. So that, these samples could be added to the training set to improve weak spots in the learned decision boundary. Going beyond in random erasing, in [130], they train an adversarial network in parallel with the detector Fast-RCNN to generate examples with occlusions and deformations that would lead the detector to misclassification. The Fast-RCNN will adapt itself to learn to classify these adversarial examples and be, in turn, more robust invariant to occlusions and deformations.

The above principles of adversarial training have led to the popular generative adversarial networks (GANs) firstly presented by Goodfellow *et al.* [42]. The model consists of two networks that are trained in an adversarial process where, iteratively, one network (the generator) generates fake images and the other network (the discriminator) discriminates between real and fake images. So that, here, the adversarial loss forces the generated images to be, in principle, indistinguishable from real ones. Then, Radford *et al.* [101] applied the Goodfellow idea for generating images with the Deep Convolutional GANs (DCGAN). GANs were first introduced to address the problem of generating realistic images but, due to their great potential, they have gained great popularity in the computer vision community and different variations of GANs were recently proposed for image synthesis [57, 104, 152], image super-resolution [76], image inpainting [144], or image blending [134], among others.

Image super-resolution. Image super-resolution comprises the task of estimating a high resolution (HR) image from its low-resolution (LR) counterpart. The techniques to achieve the final image can use a series of consecutive frames of a video or a single image. Multiple image-based (or classical) solutions are mostly reconstruction-based algorithms that try to address aliasing artefacts by simulating the image formation model [7, 31]. These models are highly dependent on the motion estimation between the LR images, so they are more unstable in real-world applications [92]. Henceforth, we will describe only single image super-resolution (SISR) approaches.

Before the emergence of convolutional neural networks, SISR techniques ranged from simple prediction-based methods (bilinear, bicubic, nearest neighbor...), which yield solutions with overly smooth textures, going through methods that attempt to address these shortcomings by exploiting different priors. In [138], they group these solutions into edge-based [36, 117], statistical [67, 136], patch-based [49, 140], or sparse dictionary methods [141].

With the remarkable CNN success, all efforts were turned in this direction. Within this scope, Dong *et al.* [25, 26] used bicubic interpolation to upscale an input image and feed a three layer deep fully convolutional network to achieve state-of-the-art SR performance. Kim *et al.* [66] achieved high performance by using a deeply-recursive convolutional network (DRCN) that allows for long-range pixel dependencies while keeping a low number of parameters. Another approach [123] proposed a CNN with skip connections where the feature maps of each layer are propagated into all subsequent layers to combine the low- and high-level features, enhancing the reconstruction performance.

The definition of a perceptual loss [59], instead of low-level pixel-wise error measures, represented a significant improvement. The perceptual loss function applies an L_2 loss over calculated feature maps using another pre-trained CNN —such as VGG— to increase the perceptual similarity, which leads to recover visually more convincing HR images. More recently, GANs boosted even more the image super-resolution results. An impressive work was done in [76] where they proposed SRGAN, a GAN trained using the perceptual loss in cooperation with the adversarial loss to infer photo-realistic natural images for $4\times$ upscaling factors. Based on the previous study, Wang *et al.* [131] released Enhanced SRGAN (ESRGAN) by improving the network architecture, the adversarial loss and the perceptual loss.

In spite of the progress obtained with GANs, to train these networks it is necessary to have pairs of low resolution (LR) and high resolution (HR) images. Most of the approaches use bilinear interpolation to obtain the LR images, which is shown in [10, 113], but they cannot produce good results for real-world low-resolution images. To address this, Bulat *et al.* [10] defined two consecutive GANs where the first GAN learns to degrade high resolution (HR) images to low resolution (LR), images and the second GAN uses these LR images to learn the standard image super-resolution.

Image Inpainting. Image inpainting is a conservation process where damaged, deteriorated, or missing parts are filled in to present a complete image. Existing works for image inpainting can be mainly divided into two groups. Until the CNNs breakthrough, this problem was dealt with diffusion or patch-based methods with low-level features. Currently, most of the methods attempt to solve it by a learning-based approach using convolutional neural networks to predict pixels for the missing regions.

The first group of approaches often uses patch similarity algorithms to propagate information from background regions to holes [5, 28]. This method was optimized by Wexler *et al.* [133], and Simakov *et al.* [114] proposed a global patch similarity-based scheme to better capture and summarize non-stationary visual data. This dense computation of patch similarity is a very expensive operation, so these techniques were later accelerated by a randomized patch search algorithm called PatchMatch [3].

The obvious limitation with these approaches is that the synthesized texture only comes from the input image or video. Therefore, learning-based algorithms have now superseded patch-based methods as they are able to learn adaptive image features for different semantics. Initially, CNN-based image inpainting approaches were limited to very small and thin masks [70, 105]. In the same way as in image super-resolution, the establishment of the GANs has lead to better inpainting results, as the discriminator forces the generator to fill with coherent data within the dataset. Specifically, Pathak *et al.* [96] introduced a Context Encoder (CE) trained with both L2 pixel-wise reconstruction loss and generative adversarial loss as the objective function to complete large center regions of fixed size. In [55], they extended the previous work by introducing both global and local discriminators as adversarial losses, and a fully convolutional network to handle arbitrary resolutions and multiple holes.

More recently, Yu *et al.* [143] propose a novel contextual attention layer to borrow features from distant spatial locations during training to improve the final performance. In [81], they propose partial convolutions to address the deficiency of standard convolutions when dealing with free-form holes. Based on the two preceding analyses, Yu *et al.* [144] released a generative image inpainting system using gated convolutions as a learnable dynamic feature selection mechanism for each channel at each spatial location across all layers.

Image blending. Image blending refers to paste a foreground region from a source image into the target background at a specified location, where the goal is to improve the spatial and color consistencies to make the composite image look as natural as possible. The default way of doing this task is to directly copy pixels from the source image and paste them onto the target image, but this would generate obvious artefacts because of the abrupt intensity change in the compositing boundaries.

Alpha blending [100] is a simple and fast process that overlays a foreground image with transparency over a background image. Even though the results are better than copy and paste, it blurs the fine details when there are some registration errors between the source and target images and produces the ghost effect when the alpha window is not optimal. Burt and Adelson [11] introduced Laplacian pyramid, a multiresolution representation of the images of interest. The source images are decomposed into a set of band-pass filtered component images, then joined within each resolution band independently and, finally, adding up the different levels. So that, when coarse features occur near borders, these are blended gradually over a relatively large distance without blurring or otherwise degrading finer image details in the neighborhood of the border.

Alternatively, the most popular image blending technique aims to inform gradient domain smoothness [99, 119, 120], which enables a smooth transition and reduces the color/illumination differences between foreground and background. In [99], they firstly produce a gradient vector field based on the gradients of the composite image, and then recover the blended image from this gradient vector field by addressing a Poisson equation. Such methods are good at generating high-resolution results with rich details and textures but the generated images tend to be unrealistic, as they contain various kinds of artefacts.

A recent approach GP-GAN [134] has leveraged the closed-form solution of the Gaussian-Poison equation by training a GAN to produce photo-realistic blending results. However, this approach relies on supervised training, which requires paired data of a source image, a target image, and the corresponding well-blended image as ground-truth, so the generalization is difficult.

4.3 Small object data augmentation

Figure 4.2 shows the architecture of the pipeline for data augmentation for small object detection designed in this PhD Thesis. The purpose of this architecture is to increase the number of small objects in a video dataset. Our system consists of two procedures: the *small object generation*, which involves object downsampling and object segmentation, and the *small ob-*

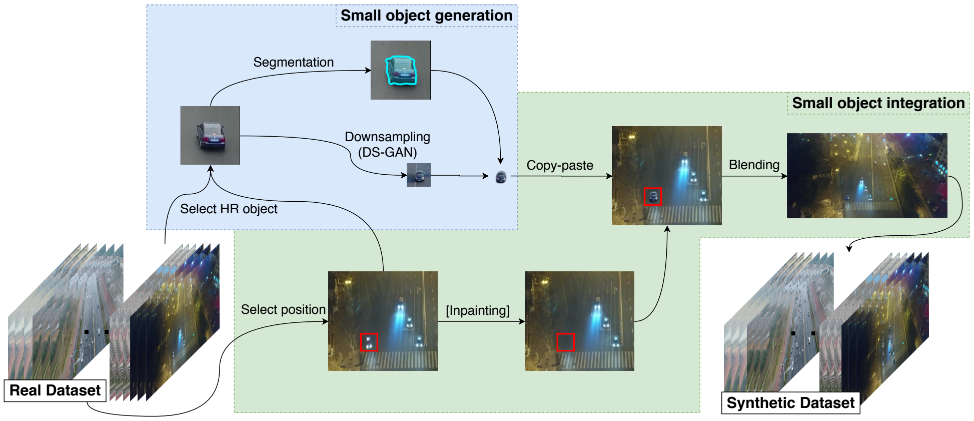


Figure 4.2: The proposed pipeline for data augmentation for small object detection. It takes a video dataset and produces the same frames but populated with synthetic small objects. The system comprises two main steps: small object generation and integration. This is repeated for each position/HR object pair.

ject integration into the image, which involves position selection, object inpainting and object blending.

Through these components the system is able to generate synthetic low resolution (SLR) objects from real high resolution (HR) objects; these SLR objects will have similar features to real low resolution objects (LR). Then, they are inserted in plausible positions within the image without looking for a temporal consistency between frames. The following are the steps performed by the pipeline applied to an input video dataset (Figure 4.2):

- The *small object generation* procedure produces SLR objects and their corresponding masks from HR objects.

  1. The **object downsampling** generates an SLR object from an HR object with its context.
  2. The **object segmentation** calculates the input HR object segmentation mask and transforms it to fit the SLR object.

- The *small object integration* procedure selects the optimal positions for the SLR object and inserts it into the image.

1. The **position selector** selects the possible positions where some real low resolution (LR) objects exist —or existed in previous or successive frames— and optimizes the position and SLR object matching by comparing the direction and shape of both LR and HR objects through optical flow and overlap.
2. The **object inpainting** deletes the objects that will be replaced.
3. The **object blending** makes a copy-paste of each SLR object in the matched position and performs a blending operation to alleviate the abrupt boundary change and color intensity on the scene.

The final result provided by our system is a new dataset created with the same video images but populated with an increased number of SLR objects that replace the fixed number of LR objects.

4.3.1 Small object generation

Downsampling GAN (DS-GAN)

A straightforward solution for obtaining SLR objects from large objects is the use of a rescaling function such as the well-known bilinear interpolation or nearest neighbor. However, as indicated in Section 4.2 and proved in the experimentation (Section 4.4), the use of this kind of function introduces artefacts that produce objects with a very different features statistical distribution, which make them useless for data augmentation. Thus, to generate useful SLR objects from a learning perspective for object detection, a more elaborated system is required. For this task, a Downsampling GAN (DS-GAN) has been designed. DS-GAN is a generative adversarial network that learns to correctly degrade HR objects into SLR objects to increase the training set for object detection.

In this downsampling problem the aim is to estimate an SLR object from an input HR object with a downsampling factor r. The problem to solve is an unpaired problem where HR objects do not have a corresponding LR pair, but the network would have to learn the features of the whole LR subset while keeping similar visual appearance of the original HR object. For an image with C color channels, HR has size $W \times H \times C$ while both LR and SLR are described by $\frac{W}{r} \times \frac{H}{r} \times C$. So, for training the proposed GAN, two different image sets are required: (i) the *HR subset* composed of real large objects (HR objects) and (ii) the *LR subset* composed of real small objects (LR objects). Both the LR and HR subsets can be taken from the same dataset or from any additional one if more samples are needed.

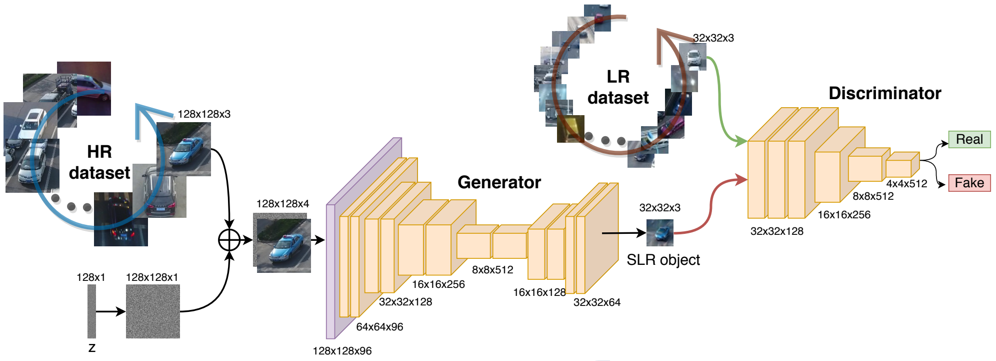


Figure 4.3: Downsampling Generative Adversarial Network (DS-GAN) architecture. The generator is trained with HR objects to synthesize small objects. A discriminator between real and fake small objects forces the generator to produce synthetic objects that are increasingly similar to real-world small objects.

Our DS-GAN architecture is shown in Figure 4.3. It is based on the High-to-Low part of the GAN presented in [10]. The generator network (G) takes as input an HR image concatenated with a noise vector (z) and produces an SLR image $4\times$ smaller than the input ($r = 4$). For example, a 128×128 object will lead to a 32×32 object. The noise vector is randomly sampled from a normal distribution and it is attached to the input image to model the fact that the HR image will be affected by multiple types of LR noise. This allows to produce numerous SLR objects from a single HR object. Following the methodology of [42] we further define a discriminator network (D) which we optimize in an alternating manner along with the generator (G).

The generator is an encoder-decoder network —see Figure 4.3— composed of six groups of residual blocks [47]. Each group has two same-dimension residual blocks with pre-activation and batch normalization as defined in [48]. To achieve a $4\times$ downscaling, four $2\times$ down-sample steps performed by pooling layers are placed at the end of each of the first four groups and two $2\times$ up-sample steps performed by deconvolution layers at the end of each of the last two groups.

The discriminator —see Figure 4.3— follows the same residual block structure (without batch normalization) followed by a fully connected layer and a sigmoid function. The discriminator is composed of six residual blocks with two $2\times$ down-sample steps performed by pooling layers at the end of the last two blocks. The details of the composition of both architectures are better shown in Figure 4.3.

With this architecture, our goal is to train G to generate a synthetic low resolution sample (SLR) conditioned on a HR sample. To achieve this, the objective function chosen for the adversarial loss is the hinge loss [87]:

$$l_{adv}^{D} = \mathop{\mathbb{E}}_{s \sim \mathbb{P}_{LR}} [min(0, 1 - D(s))] + \mathop{\mathbb{E}}_{\hat{s} \sim \mathbb{P}_{G}} [min(0, 1 + D(\hat{s}))], \tag{4.1}$$

where $\mathbb{P}_{LR}$ is the LR subset distribution and $\mathbb{P}_{G}$ is the generator distribution to be learned through the alternative optimization. $\mathbb{P}_{G}$ is defined by $\hat{s} = G(b,z) \mid b \in \mathbb{P}_{HR}$, where $\mathbb{P}_{HR}$ is the HR subset. The general idea behind this formulation is that it allows to train G with the goal of fooling D, that is trained to distinguish SLR from LR images. With this approach our generator can learn to create SLR solutions that are highly similar to real LR images, and thus difficult to classify by D.

Correspondingly, we train G by optimizing a loss function $\mathscr{L}$, which is defined as:

$$\mathscr{L} = l_{pixel} + \lambda l_{adv}^{G}, \tag{4.2}$$

where l_{adv}^{G} is the adversarial loss, l_{pixel} is the L_2 pixel loss, and λ is a parameter that balances the weight of both components.

The adversarial loss l_{adv}^{G} is defined based on the probabilities of the discriminator as:

$$l_{adv}^{G} = - \mathop{\mathbb{E}}_{b \sim \mathbb{P}_{HR}} [D(G(b,z))], \tag{4.3}$$

where $\mathbb{P}_{HR}$ is the HR subset and z is the noise vector. The adversarial loss is computed in an unpaired way, using the LR subset to make the SLR objects to be contaminated with real-world artefacts.

The l_{pixel} minimizes the L_2 distance between the input HR and the output SLR:

$$l_{pixel} = \frac{r^2}{WH} \sum_{i=1}^{\frac{W}{r}} \sum_{j=1}^{\frac{H}{r}} (AvgPooling(b)_{i,j} - G(b,z)_{i,j}) \mid b \in \mathbb{P}_{HR}, \tag{4.4}$$

where W and H denote the input HR size, r is the downsampling factor and $AvgPooling$ is an average pooling function that maps the HR input to the output $G(b,z)$ resolution. The l_{pixel} is computed in a paired way between the SLR object and the HR object downsampled to the output SLR resolution using an average pooling layer. This component aims to keep the appearance of the synthetic objects similar to the original HR objects.

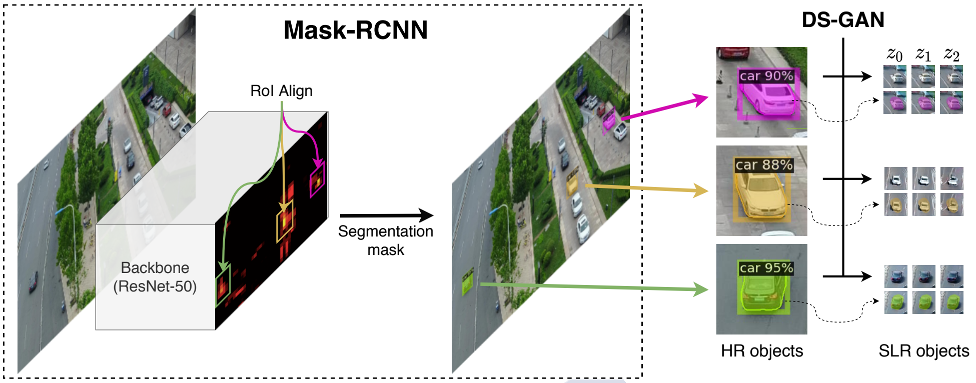


Figure 4.4: HR object segmentation using the Mask R-CNN framework [46] (right), and DS-GAN outputs for different noise vectors (z_i) with the masks fitted to the SLR objects (left).

In addition, to solve the stabilization of the discriminator training we normalize its weights by the spectral normalization technique [87].

Object Segmentation

The approach chosen for object segmentation is to adapt the Mask R-CNN framework [46] trained on the public dataset MS COCO to obtain the mask from HR objects (Figure 4.4). As the segmentation results for small objects have a poor performance [90], we propose to get the mask from the large objects and fit it to the small objects. This adaptation is done just by resizing by factor r. This is possible because the pixel loss (Equation 4.4) forces the generator to keep the visual object appearance, i.e., pose, orientation, size, etc. Figure 4.4 shows the masks adaptability from HR to SLR objects.

Adding this process solves three issues: (i) the pipeline does not limit its performance to the existence of objects with a mask ground truth, which is missing in many popular datasets [9, 27, 153] as the annotation is very costly; (ii) the small object segmentation is optimized, as the performance of segmentation methods declines dramatically for small objects [90]; and (iii) there is no need to use the SLR objects to generate the segmentation mask —SLR objects do not contain enough context to get a proper mask (Figure 4.4).

4.3.2 Small object integration

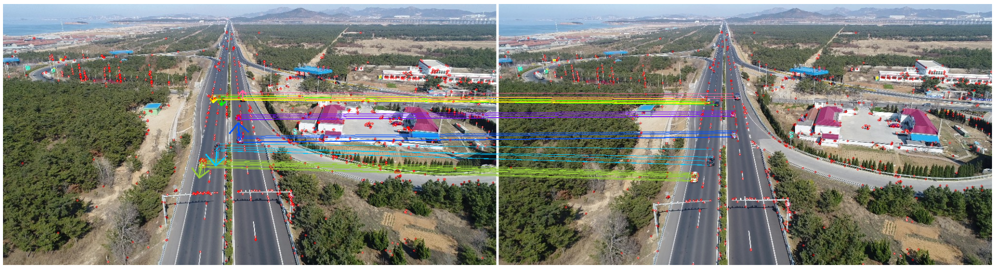


Figure 4.5: Motion angle estimation using optical flow for two frames f_t (right) and f_{t+1} (left). First, the feature points are computed using FAST (red dots). Second, f_{t+1} is stabilized with f_t by perspective transformation to remove camera motion. Then, the feature points are matched between frames (colored lines). Finally, the motion lines are summarized into a motion vector for each object (colored arrows).

Position selector

The selection of a position within the image is a key issue when performing data augmentation for object detection. If this position is randomly selected, the new context surrounding the objects could be counterproductive, i.e., background mismatch may lead the model to generate more false-positive bounding boxes. The reason is that the detector learns on not only the object features but also the context features, using the background prior knowledge to assist itself [15].

In order to sample a suitable position according to the image background, three premises must be fulfilled: (i) to have a plausible background —e.g., a car must be placed into the road—; (ii) the orientation has to fit the scene —e.g., a car's orientation has to match the direction of the road—; and (iii) the scale has to be according to the vanishing point of the frame —p.e. small objects cannot be placed in the foreground. Therefore, to cover these requirements, our proposed position procedure is also based on three techniques: spatial memory of the objects to obtain a plausible background, optical flow to match orientations, and overlap to match scales. As pointed out above, no temporal consistency for objects between frames is demanded; they only need to have spatial significance within the frame.

The spatial memory of the objects aims to collect plausible positions where to place an SLR object in the current frame. This method is based on selecting in the current frame the LR object position in previous and subsequent frames. These positions can be taken as valid as long as there is no object in the current frame.

Optical flow and overlap aim to pair each candidate position with the SLR object that most closely resembles the orientation and size. We exploit optical flow to compute the apparent motion of objects within two frames (Figure 4.5): (i) we detect FAST keypoints [108]; (ii) stabilize camera motion by perspective transformation; (iii) link feature points between f_{t-1} and f_t within each bounding box by optical flow; (iv) compute the motion angle for each object in f_t by averaging all its points into a motion vector. The overlap between two objects is computed via Intersection over Union (IoU).

Given the motion angle and the dimensions associated to the HR and LR objects, each possible position gets this information from the LR object from which it has given rise and each SLR object from its original HR object. Then, each position and SLR object pairing will be given by maximizing the overlap and motion angle similarity between them.

Algorithm 2 shows the position selector method for each video:

- **Input**: The algorithm takes as input the total set of objects in the dataset (GT) within each frame f at time t (f_t) —which includes the LR and HR subsets—, the total set of SLR objects obtained by the DS-GAN generator G from HR objects and the search range τ.

- **Ouput**: The algorithm returns the association ($\mathbb{A}$) of an SLR object ($\hat{s}_i$) for each empty space (e_j) —$\hat{s}_i$ can be linked to more than one e_j.

- **Spatial memory** (lines 4-17): Given frame f at time t, the possible empty spots (E_t) to place an SLR object ($\hat{s}_i$) will be those where an LR object (s_j) existed in the frames from $f_{t-\tau}$ to $f_{t+\tau}$ (line 4) —s_i^t is always valid (line 6). For each frame $f_{t'}$ of the interval ($f_{t-\tau}$, $f_{t+\tau}$) the algorithm checks if the $LR^{t'}$ objects overlap with any of the objects of the current frame (GT^t) or with any space already selected (E^t) (lines 9-15). Otherwise, $s_i^{t'}$ is added as new empty spot to E^t (line 17). Thus, each possible empty spot e_j^t corresponds to a position of an LR object ($s_i^{t'}$).

  The value of τ will be influenced by the video dataset and, more specifically, by the camera motion. The more the camera moves, the less the value of τ will be to avoid background mismatch. If the camera motion is too quick, the positions of the objects in previous or subsequent frames may correspond to an erroneous position in the image –e.g., a car on a sidewalk.

Algorithm 2: Position selector

```
Input  : $GT = \{GT^t = \{g_1^t, \dots, g_{n_t}^t\}\ \forall\, t = 1, \dots, T\}$
Input  : $LR = \{LR^t = \{s_1^t, \dots, s_{m_t}^t\}\ \forall\, t = 1, \dots, T\} \mid LR \in GT$
Input  : $HR = \{b_1, \dots, b_l\} \mid HR \in GT$
Input  : $SLR = \{\hat{s}_1, \dots, \hat{s}_l\} \mid \hat{s}_i = G(b_i, z)\ \forall\, i = 1, \dots, l$
Input  : Search range $\tau$
Output: $\mathbb{A} = \{\mathbb{A}^t = \{(e_i^t, \hat{s}_{k(i)}) \dots (e_n^t, \hat{s}_{k(n)})\}\ \forall\, t = 1, \dots, T,\ e_i^t \in E_t\}$
$\mathbb{A} \leftarrow \emptyset$
for $t = 1, \dots, T$ do
    $E_t \leftarrow \emptyset$
    for $t' = \mathtt{max}(0, t-\tau), \dots, \mathtt{min}(T, t+\tau)$ do
        if $t = t'$ then
            $E_t \leftarrow E_t \cup s_i^t$
        else
            for $i = 1, \dots, m_{t'}$ do
                $valid\_spot = 1$
                for $j = 1, \dots, n_t$ do
                    if $\mathtt{IoU}(s_i^{t'}, g_j^t) > 0$ then
                        $valid\_spot = 0$
                for $j = 1, \dots, \mathtt{size}(E_t)$ do
                    if $\mathtt{IoU}(s_i^{t'}, e_j^t) > 0$ then
                        $valid\_spot = 0$
                if $valid\_spot = 1$ then
                    $E_t \leftarrow E_t \cup s_i^{t'}$
    for $i = 1, \dots, \mathtt{size}(E_t)$ do
        $max_v = max_j = 0$
        $\alpha_i^t = \mathtt{OpticalFlow}(e_i^t)$
        for $j = 1, \dots, l$ do
            $\alpha_j = \mathtt{OpticalFlow}(b_j)$
            $\Delta_{i,j} = 1 - \mathtt{normalize}(|\alpha_i^t - \alpha_j|)$
            $iou_{i,j} = \mathtt{IoU}(e_i^t, \hat{s}_j)$
            if $\Delta_{i,j} + iou_{i,j} > max_v$ then
                $max_v = \Delta_{i,j} + iou_{i,j}$
                $max_j = j$
        $\mathbb{A}^t \leftarrow \mathbb{A}^t \cup (e_i^t, \hat{s}_{max_j})$
    $\mathbb{A} \leftarrow \mathbb{A} \cup \mathbb{A}^t$
```

- **Object association** (lines 18-28): The best $\hat{s}_i$ is calculated for each of the empty spots e^t_j by maximizing the motion direction and overlap.
  - **Optical flow**: For each ground truth LR and HR objects in the video dataset, an angle (α) associated with its motion vector is pre-calculated through optical flow (Figure 4.5) —lines (20 and 22). As in the segmentation step, the $\hat{s}_i$ motion vector can be derived from its original HR object b_i (OpticalFlow$(\hat{s}_i)$ = OpticalFlow(b_i)). Considering the SLR and the LR subsets, motion similarity (Δ) associated with each pair $\hat{s}_i$, s_j is given by:

$$\Delta_{i,j} = 1 - \text{normalized}|\text{OpticalFlow}(b_i) - \text{OpticalFlow}(s_j)| \tag{4.5}$$

  - **Overlap**: Likewise, the $\hat{s}_i$ size can be derived from its original HR object b_i ($w(\hat{s}_i) = \frac{w(b_i)}{r}$; $h(\hat{s}_i) = \frac{h(b_i)}{r}$). Then, the overlap between $\hat{s}_i$ and s_j is computed using Intersection over Union (IoU).

Finally, the i-th SLR object selected to fill the position e^t_j will be given by:

$$i = \max(\Delta_{i,j} + \text{IoU}(\hat{s}_{k(j)}, e^t_j)) \; \forall \, t = 1, \ldots, T, \; e^t_j \in E_t, \tag{4.6}$$

Inpainting

The position selector procedure considers each s^t_j in f_t as an empty spot e^t_j for filling with $\hat{s}_i$. In these situations, it is mandatory to remove s^t_j associated to e^t_j before inserting its pair. To this end we perform image inpainting using DeepFill [143]. DeepFill is a generative model-based approach which can synthesize novel image structures using surrounding image features.

Deepfill takes as input the frame f_t and a mask m_t and returns the same image f'_t but with the empty regions filled. For generating the mask m_t associated with the frame f_t, the bounding boxes of the selected LR objects $s^t_i \in E^t$ will be considered, so that those pixels contained in them will be flagged ($m_t = 1$).

The Deepfill generator architecture is sketched in Figure 4.6. The generator comprises two encoder-decoder networks for two different purposes. The first one —coarse network— aims to make an initial coarse prediction and, the second network —refinement network— takes the coarse prediction as inputs and predicts the final result f'_t. The reason for these two networks is intuitive: the refinement network sees a more complete scene than the original

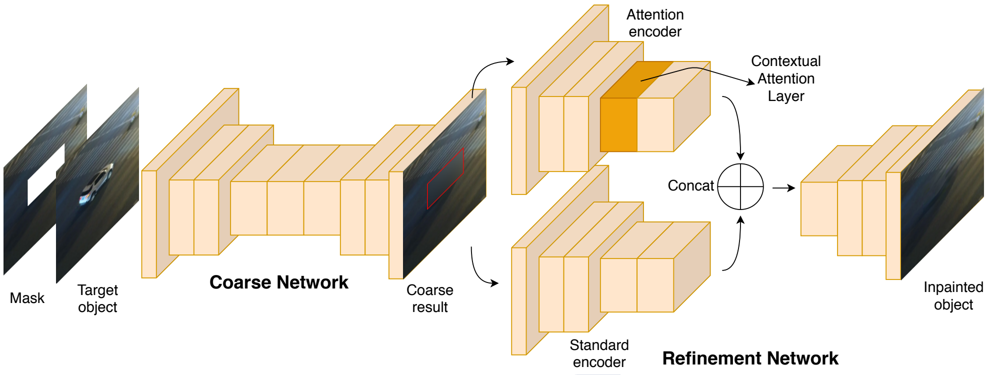


Figure 4.6: DeepFill architecture for object inpainting [143]. The network takes as input an image and a mask pointing out the object to be deleted, and returns the same input image but with the masked area filled in.

image with missing regions, so its encoder can better learn feature representation than the coarse network.

As LR^t may be surrounded by other objects, it is interesting to borrow distant image features within the image without objects. This is addressed by DeepFill with two parallel refinement network encoders concatenated at the end into a single decoder. The standard encoder specifically focuses on refining local contents with layer-by-layer (dilated) convolution, while the attention encoder tries to capture background interest features.

Insertion and Blending

As a final stage, the pipeline blends the corresponding SLR object $\hat{s}_i$ obtained by Equation 4.6 over an f_t' inpainted image obtained in the previous step in each of the spots e_j^t to generate f_t^*. The copy-paste is straightforward using the segmentation information collected in the segmentation step.

Then, the blending step is required to improve color consistencies and to soft the object edges in order to make the composite image look as natural as possible. We have adopted the Laplacian pyramid introduced by Burt and Adelson [11] to blend the SLR objects into the video frames.

This blending method takes as input an inpainted video frame f_t', the copy-pasted image f_t'' and the mask image m_t' that points out where to blend. In the inpainting stage, the flagged

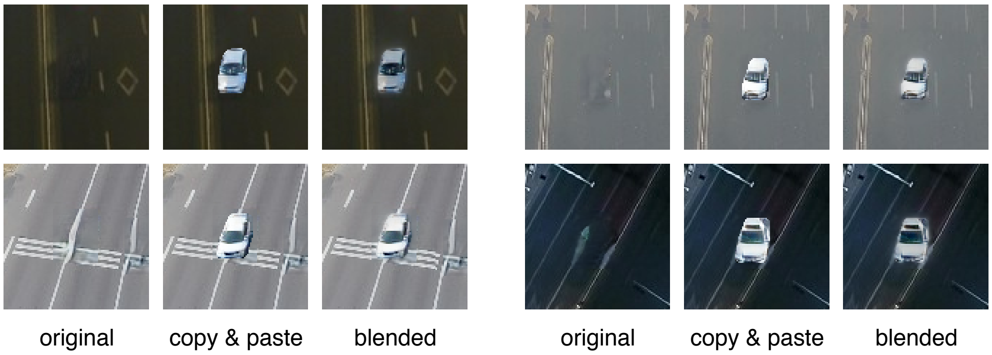


Figure 4.7: Image results of blending for similar synthetic small objects on different background conditions.

pixels in m_t are those inside the bounding box ground truth, but in m'_t the flagged pixels are those from the SLR segmented pixels. Figure 4.7 shows some examples for this final blending stage. Algorithm 3 details the procedure to obtain the final synthetic video frame:

1. Create the temporal image f''_t by copy-pasting each $\hat{s}^t_{k(i)}$ object in e^t_i on f'_t (line 4). Generate the mask m'_t by flagging those pixels that belong to $\hat{s}^t_{k(i)}$ (line 5).

2. Compute p levels of Gaussian pyramids for f'_t, f''_t and m'_t (lines 6-12). Each Gaussian pyramid level is the result of blurring and downsampling the previous one.

3. From the Gaussian pyramids, calculate the Laplacian pyramid for f'_t and f''_t (lines 13-17). Each Laplacian pyramid level is the result of subtracting each Gaussian pyramid level with the up-sampled and blurred previous one. The smaller level in the Laplacian pyramid is the same as the smaller in Gaussian pyramid.

4. Next, each level of the Laplacian pyramid is blended according to m'_t of the corresponding Gaussian level (line 21). The set of masks (M'_t) is previously reversed to match the dimensions (line 19).

5. Finally, from this blended pyramid, the output image (f^*_t) is reconstructed by up-sampling and blurring each level and adding it to the next one (line 23-26).

Algorithm 3: Insertion and blending algorithm

```
Input  : $\mathbb{A}^t = \{(e_i^t, \hat{s}_{k(i)}) \dots (e_n^t, \hat{s}_{k(n)})\}\ \forall\, t = 1, \dots, T,\ e_i^t \in E_t$
Input  : Inpainted image: $f_t'$
Input  : Pyramid levels: $p$
Output: Final synthetic image: $f_t^*$
$f_t'' \leftarrow f_t'$
$m_t' \leftarrow \emptyset$
for $i = 1, \dots, n$ do
    $f_t''[e_i^t] = \hat{s}_{k(i)}^t$
    $m_t'[\hat{s}_{k(i)}] = 1$
$F_t' \leftarrow \{f_t'\}$
$F_t'' \leftarrow \{f_t''\}$
$M_t'' \leftarrow \{m_t'\}$
for $i = 1, \dots, p$ do
    $F_t' \leftarrow F_t' \cup \texttt{PyramidDown}(F_t'[i])$
    $F_t'' \leftarrow F_t'' \cup \texttt{PyramidDown}(F_t'[i])$
    $M_t' \leftarrow M_t' \cup \texttt{PyramidDown}(F_t'[i])$
$L_t' \leftarrow \{F_t'[p]\}$
$L_t'' \leftarrow \{F_t''[p]\}$
for $i = p, \dots, 2$ do
    $L_t' \leftarrow L_t' \cup (F_t'[i-1] - \texttt{PyramidUp}(F_t'[i]))$
    $L_t'' \leftarrow L_t'' \cup (F_t''[i-1] - \texttt{PyramidUp}(F_t''[i]))$
$B_t \leftarrow \emptyset$
$M_t'' \leftarrow \texttt{Reverse}(M_t'')$
for $i = 1, \dots, p$ do
    $b = L_t'[i] \times M_t'[i] + L_t'' \times (1 - M_t'[i])$
    $B_t \leftarrow B_t \cup b$
$b \leftarrow B_t[1]$
for $i = 2, \dots, p$ do
    $b = \texttt{PyramidUp}(b) + B_t[i]$
$f_t^* \leftarrow b$
```

4.4 Experiments

In this section we address the experimentation carried out for the evaluation of the quality of the synthetic objects generated by DS-GAN, and for the final detection improvement using the pipeline for data augmentation for small object detection on state-of-the-art networks. We also define the datasets, evaluation metrics and implementation details to validate each of the experiments.

4.4.1 DS-GAN

For this experimentation, the SLR objects generated by the DS-GAN are compared with the LR objects —aiming for the greatest similarity— as well as with the resizing functions: linear interpolation, bicubic interpolation, nearest neighbours and Lanczos [124]. For this purpose, two metrics will be used to validate the quality of the synthetic objects generated by DS-GAN:

- **Frechet Inception Distance (FID)** [50]: FID is a popular metric for comparing the feature vectors calculated for real and generated images. The FID score summarizes how similar the two groups are in terms of statistics on computer vision features of the raw images calculated using a pre-trained image classification model —commonly Inception-v3 [118]. The lower the scores the greater the similarity of the two groups, meaning that they have more similar statistics, which is the purpose of our DS-GAN.

- **Object classification**: To support the above metrics, we also train an LR object classifier which differentiates between background (negative) and LR object (positive). We resort to this metric since it is closer to the objective of the full pipeline, i.e., the improvement of small object detection. On the one hand, the classifier is trained with the LR training set as positive examples and a background set as negative examples. On the other hand, the SLR set is used for positive examples and keeping the same backgrounds as negative examples. We have generated different SLR sets, one for each of the resizing functions, and another one for the DS-GAN. All the learned models are evaluated with the LR testing subset and different backgrounds. The higher the accuracy, the better the quality of the objects synthetically generated.

The DS-GAN generator architecture has a final stride $4\times$ smaller than the fixed size input image ($r = 4$). Most of the popular datasets —MS COCO [80], UAVDT [27], VisDrone [153]— consider as small objects those smaller than 32×32 pixels. Therefore, we will train

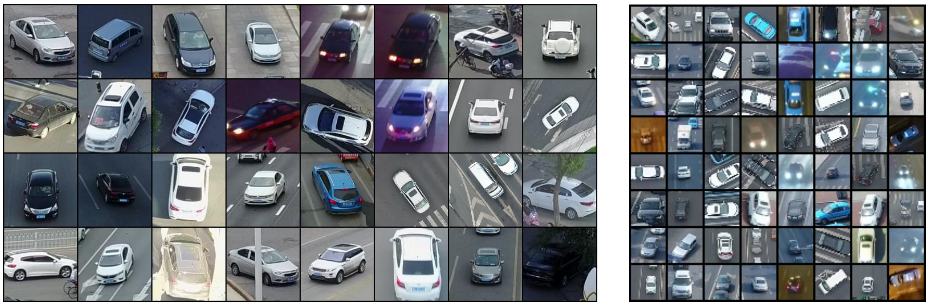


Figure 4.8: (a) Real HR samples, and (b) real LR samples from the UAVDT dataset.

the DS-GAN to learn how to reduce HR objects to that range. As examples, 128×128 and 64×64 HR objects would be 32×32 and 16×16 SLR objects provided by our DS-GAN.

We validate our data augmentation for small object detection approach with the car category on the UAVDT dataset [27]. This dataset was selected because the whole set of objects are vehicles, which allows us to isolate the results for a specific category, and also provides a large number of small instances in the testing set. Quantitatively, UAVDT comprises 23,829 frames of training data and 16,580 frames of test data, belonging to 30 and 20 videos of $\approx$ 1,024 $\times$ 540 resolution, respectively. The videos are recorded with an UAV platform over different urban areas. UAVDT includes a total of 394,633 car instances for training, where 107,091 are considered within the *small* subset (52.38%), and a total of 361,055 car instances for testing, where 274,438 are considered within the *small* subset (76.01%).

Considering that the camera motion in UAVDT slightly modifies the appearance of consecutive frames, in this section, only 10% of the video frames are selected for training to avoid overfitting. The details on the datasets for evaluating DS-GAN are given below:

- **Real high resolution (HR) subset**: To obtain the HR objects we select those objects from 48×48 to 128×128 pixels, and we add context to have an area of 128×128 pixels in objects with a smaller area. These conditions result in a total number of 517 HR objects in the UAVDT dataset. To have a larger number, we also select the cars in the VisDrone dataset with the same restrictions. VisDrone is a dataset with a very similar nature to that of UAVDT, i.e., high-resolution videos recorded with UAVs. The total number of HR objects is 5,731 after joining both datasets. Some HR examples are shown in Figure 4.8a.

- **Real low resolution (LR) training subset**: To obtain the LR objects we select those objects under 32×32 with sufficient context to cover an area of 32×32 pixels. This results in a total of 18,901 objects coming from the UAVDT training set —these objects are a part of the UAVDT *small* subset, where redundant instances have been discarded. However, in order to simulate a small object scarcity scenario, the LR subset will only consist of approximately 25% of the videos of the UAVDT dataset. The selected videos include a total of 5,226 LR objects. Some LR examples are shown in Figure 4.8a.

- **Real low resolution (LR) testing subset**: To evaluate the performance DS-GAN and the pipeline we use the 274,438 small objects coming from the UAVDT testing set with sufficient context to cover an area of 32×32 pixels.

For training the DS-GAN, we augment the training data by applying random image flipping to increase diversity. We provide a different noise vector (z) sampled from a normal distribution to each HR object in order to simulate a large variety of image degradation types. DS-GAN is trained during 1,000 epochs with an update ratio 1:1 between the discriminator and the generator, and it is optimized with Adam [68] with parameters $\beta_1 = 0$ and $\beta_2 = 0.9$. We set the base learning rate to 1e-4, decreasing it twice during the training phase by a factor of 10. We use $\lambda = 0.01$ in Eq. 4.2 to balance the relevance of the two components in the image generation process —l_{adv}^{G} is two orders of magnitude higher than l_{pixel}. Thus, the adversarial loss helps to learn to contaminate the HR input with noise and artefacts coming from the LR subset, and the pixel loss helps to preserve the visual features from the original input.

Figure 4.9 shows the experimental results to evaluate the quality of the synthetic objects generated by DS-GAN over the LR testing subset of UAVDT. Our approach is compared to the main re-scaling functions: linear and bicubic interpolation, nearest neighbors and Lanczos [124]. The reference values are obtained by the models trained on the LR training subset (blue bars).

The FID value in Figure 4.9a is measured using the final average pooling features in Inception-v3. The reference value of the LR training objects compared with the LR testing subset is 27.62. The graph of Figure 4.9a shows how the small objects obtained by any re-scaling function lead to values above 100, which is a poor performance relative to the reference value. The FID value of the SLR objects generated by DS-GAN for the LR test objects is 45.15. This FID value shows how the objects generated by the DS-GAN have better

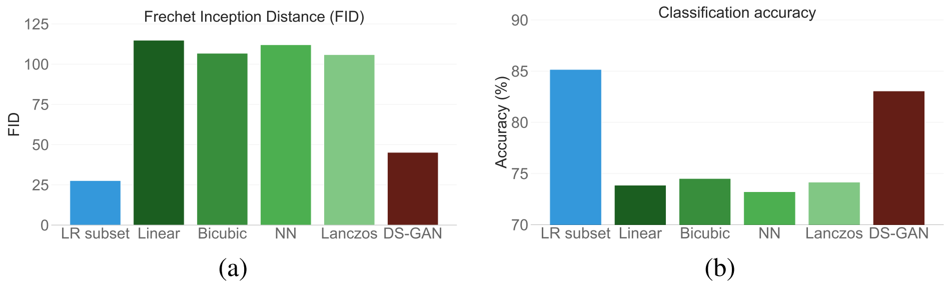


Figure 4.9: Evaluation metrics for different subsampling methods on the LR testing subset of UAVDT. (a) Frechet Inception Distance (FID) and (b) Classification accuracy. The lower the better for the FID, whereas the higher the better for the classification accuracy.

quality than those obtained by a simple re-scaling function, i.e., are more similar to the real ones.

To complement the FID distance, we have trained a classification network (ResNet-50 pre-trained on ImageNet [110]) with each of the defined subsets and tested them with the LR testing subset. Figure 4.9b shows, again, how the SLR object generated by DS-GAN provides a considerably higher accuracy (83.06%) than the re-scaling functions (≈74%), and are very close to the reference accuracy obtained by the LR training subset (85.16%).

These results validate the conclusions reached in [10, 113], since re-scaling functions introduce artefacts that make the output object differ considerably from real-world objects. Even though these differences are not visually appreciable —as we will see in Figure 4.12a below—, they are identified by the layers within the CNNs (Inception-v3 and ResNet-50). DS-GAN significantly improves this issue by learning the different artefacts found in real-world objects.

4.4.2 Data augmentation pipeline

In order to evaluate our pipeline for data augmentation for small object detection, shown in Figure 4.2, we use the UAVDT detection metrics that were originally defined by the MS COCO dataset. These metrics are the Average Precision ($AP^{@.5}$), which gives the average precision of those objects detected with at least 50% IoU between the detected and the ground-truth bounding boxes, and $AP^{@[.5,.95]}$, which is the average AP when the IoU goes from 50% to 95% in 5% steps. STDnet [9] and FPN [78] are adopted as the baseline detection networks.

Data augmentation	FPN		STDnet	
	$AP_s^{@.5}$	$AP_s^{@[.5,.95]}$	$AP_s^{@.5}$	$AP_s^{@[.5,.95]}$
LR	39.0	17.6	41.2	19.0
LR + Interp.	38.1	16.5	38.8	16.9
LR + SLR	46.3	20.1	48.1	20.6
LR + SLR×6	50.9	22.5	51.5	23.4

Table 4.1: Comparison of several data augmentation approaches for small object detection with FPN and STDnet networks on the small object testing subset of UAVDT. The training phase was conducted by simulating a low instance small object scenario —25% of the UAVDT training videos.

The implementation details for DS-GAN are those defined in the previous section. The other component that requires training is DeepFill for image inpainting. In this case, the default parameters [143] are used to train the model on the UAVDT dataset. We have set $\tau = 40$ as the frame search range for the position selector. The rest of the components of the pipeline shown in Figure 4.2 are also configured with their default values.

We detail the results obtained by STDnet [9] and FPN [78] on the UAVDT testing set for small objects. The training phase for both models was conducted from the same 25% of the videos as in the DS-GAN training, in order to simulate a scenario with a low number of LR objects, up to the whole UAVDT training set. Here, the *LR* label means that no data augmentation has been applied for training, so the images come directly from the standard UAVDT training set. The *LR + Interp.* and *LR + SLR* labels mean the same images with real objects as in *LR*, and also duplicating those images replacing the real LR objects with synthetic objects ones generated with the pipeline using bilinear interpolation and DS-GAN, respectively. So that, in *LR + Interp.* and *LR + SLR*, the number of synthetic objects is equal to the number of LR objects. Finally, the *LR + SLR*$\times n$ labels mean that the number of SLR objects is *n* times higher than the number of LR objects.

Table 4.1 studies the influence of different data augmentation methods for a scenario where the number of small objects for the training phase is reduced. So that, the first row refers only to the use of real objects contained in the 25% of the videos. The use of data augmentation with DS-GAN improves the performance of FPN by 4.9% $AP_s^{@[.5,.95]}$ and 11.9% $AP_s^{@.5}$, and STDnet by 4.4% $AP_s^{@[.5,.95]}$ and 10.3% $AP_s^{@.5}$ —Table 4.1, rows 1 and 4. It should be noted that the greatest influence is given by the nature of synthetic objects. If they did not contain useful information for learning the model, they would not improve the per-

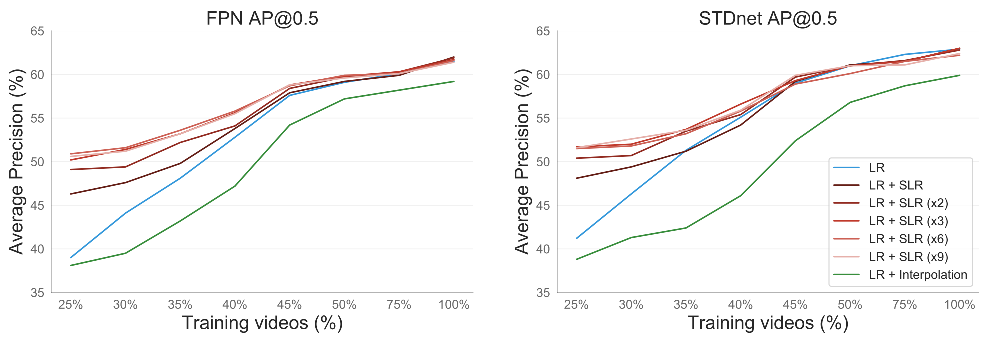


Figure 4.10: $AP_s^{@.5}$ for small object detection in UAVDT for different percentage of training videos with the FPN and STDnet architectures.

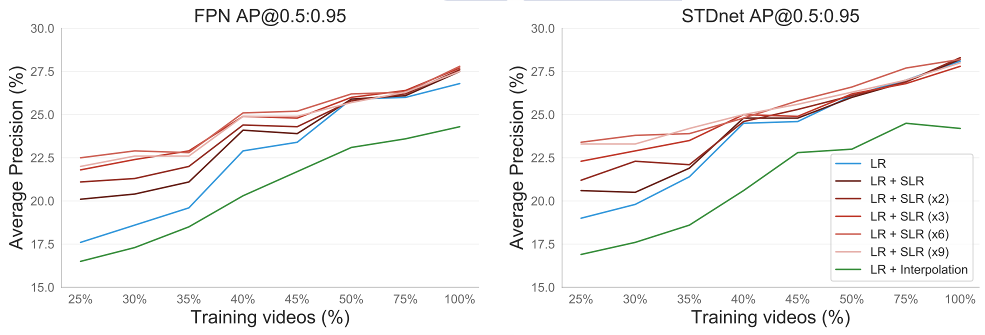


Figure 4.11: $AP_s^{@[.5,.95]}$ for small object detection in UAVDT for different percentage of training videos with the FPN and STDnet architectures.

formance, or even worsen it, as seen with the bilinear interpolation method in Table 4.1. The improvement from data augmentation with objects re-scaled by bilinear interpolation to synthetic objects generated by DS-GAN is of 3.6% $AP_s^{@[.5,.95]}$ and 8.2% $AP_s^{@.5}$ in FPN and of 3.7% $AP_s^{@[.5,.95]}$ and 9.3% $AP_s^{@.5}$ in STDnet —Table 4.1, rows 2 and 3.

Figure 4.10 and Figure 4.11 detail the extended results for the use of a different percentage of videos in the training phase and, also, show how AP changes by increasing the number of SLR objects $\times n$ in the training phase. These graphs are designed to show the improvement due to data augmentation for different percentages of training videos —with real LR objects. It is possible to appreciate a great improvement in AP for those solutions based on our data augmentation approach —the greater the number of SLR, the greater the improvement— especially when the percentage of training videos is low. As the percentage of training videos increases, the improvement is reduced, as there are more real objects in the training set. From the use of 50% of the videos onwards the AP shows a smaller improvement rate, so does the gain by adding SLR objects. That is, when adding more training images with real objects performance does not improve, and thus it is useless to try to use data augmentation techniques.

As expected, as training examples increase, so does AP. However, as mentioned above, the improvement from 50% of videos is considerably lower, moving from 59.1% $AP_s^{@.5}$ and 25.9% $AP_s^{@[.5,.95]}$ for 50% of the videos to 61.5% $AP_s^{@.5}$ and 26.8% $AP_s^{@[.5,.95]}$ for FPN on the whole UAVDT training set (blue lines, left). Similarly, the performance of the trained model with data augmentation increases as objects are added, but the gain over the baseline is lower above 50% of training videos. The same conclusions can be drawn in the case of STDnet (right).

Moreover, the models are able to take advantage of the increasing number of SLR objects until reaching a point where the progression stops —$9\times$ with respect to LR objects. To synthesize new objects above SLR$\times 3$ requires to triplicate the images and exchange the synthetic objects, because there are not enough empty spots available where to insert SLR objects. This decreases the context variability, and thus the performance improvement.

Finally, we want to highlight how the generated synthetic objects constantly improve the performance even for the complete training set (100%), where they improve $AP_s^{@[.5,.95]}$. In contrast, the objects generated by bilinear interpolation do not provide information, and even they harm the learning of the models (green lines). This confirms the high quality of the synthetic dataset produced by our pipeline for data augmentation for small object detection.

4.4.3 Qualitative results

In this section, we show different qualitative results of the DS-GAN and the full pipeline for data augmentation for small object detection. Figure 4.12 compares the synthetic objects coming from a simple re-scaling function with those generated by the DS-GAN and with the real LR objects. Objects obtained by simple re-scaling seem artificially defined with blurry artefacts. Objects from DS-GAN look more closely to real LR objects as they contain artefacts and are contaminated by low resolution small objects features. Figures 4.13-4.16 display the outcome of the complete pipeline for different UAVDT scenarios, as well as for a different number of synthetic objects.

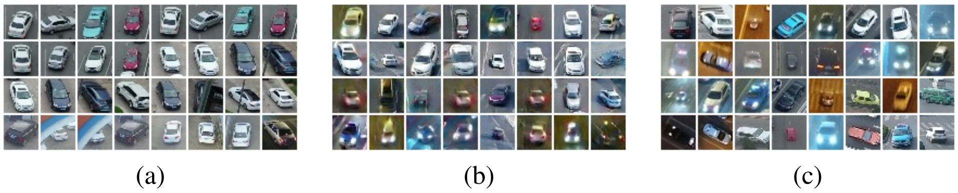


Figure 4.12: (a) Synthetic objects obtained by bilinear interpolation; (b) synthetic objects generated by the DS-GAN; and (c) real LR objects.

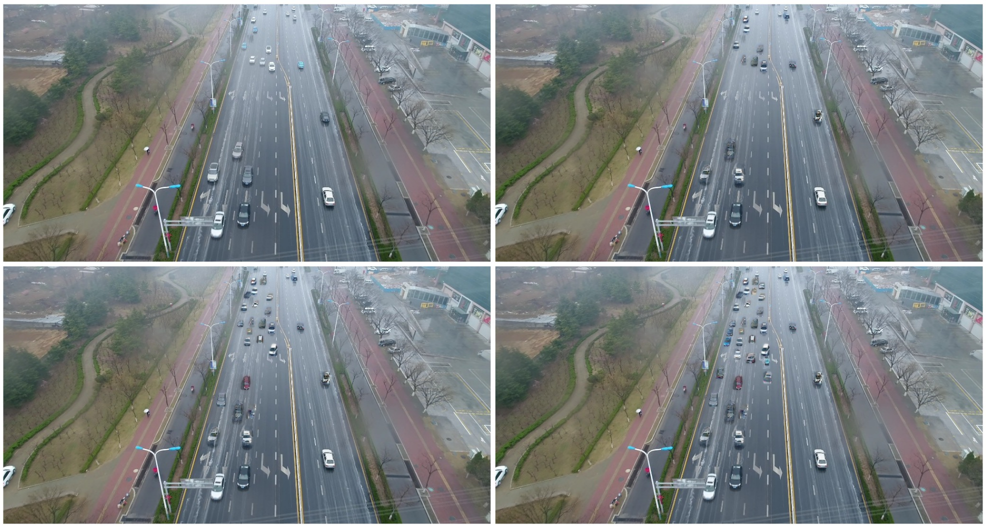


Figure 4.13: Data augmentation for small objects examples from UAVDT training set provided by our pipeline. From left to right and from top to bottom: standard real frame with LR objects; LR objects replaced by SLR objects; SLR×2 objects; and SLR×3 objects.

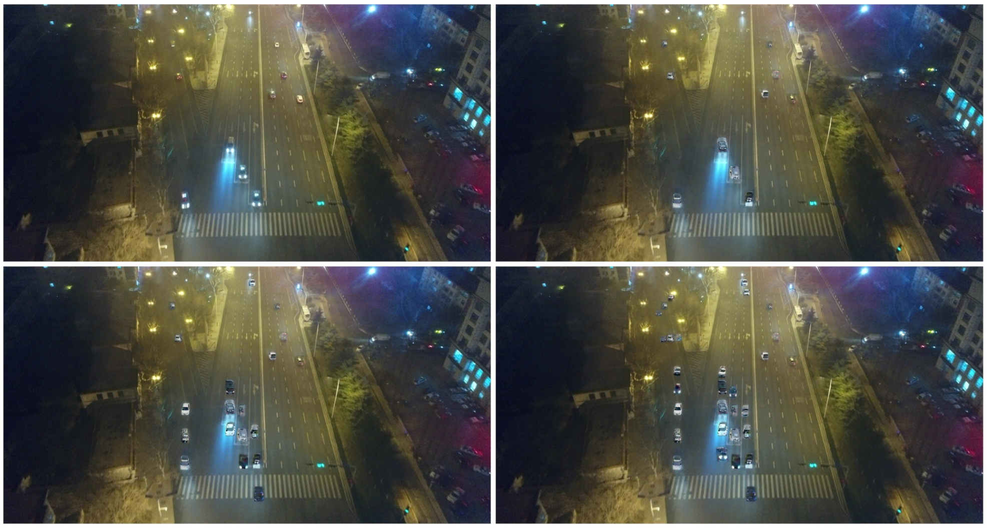


Figure 4.14: Data augmentation for small objects examples from UAVDT training set provided by our pipeline. From left to right and from top to bottom: standard real frame with LR objects; LR objects replaced by SLR objects; SLR×2 objects; and SLR×3 objects.

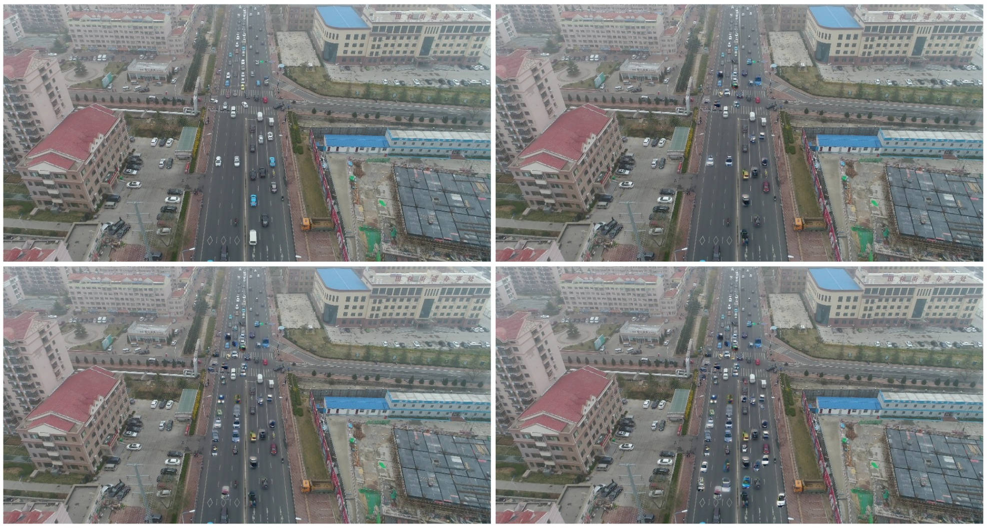


Figure 4.15: Data augmentation for small objects examples from UAVDT training set provided by our pipeline. From left to right and from top to bottom: standard real frame with LR objects; LR objects replaced by SLR objects; SLR×2 objects; and SLR×3 objects.

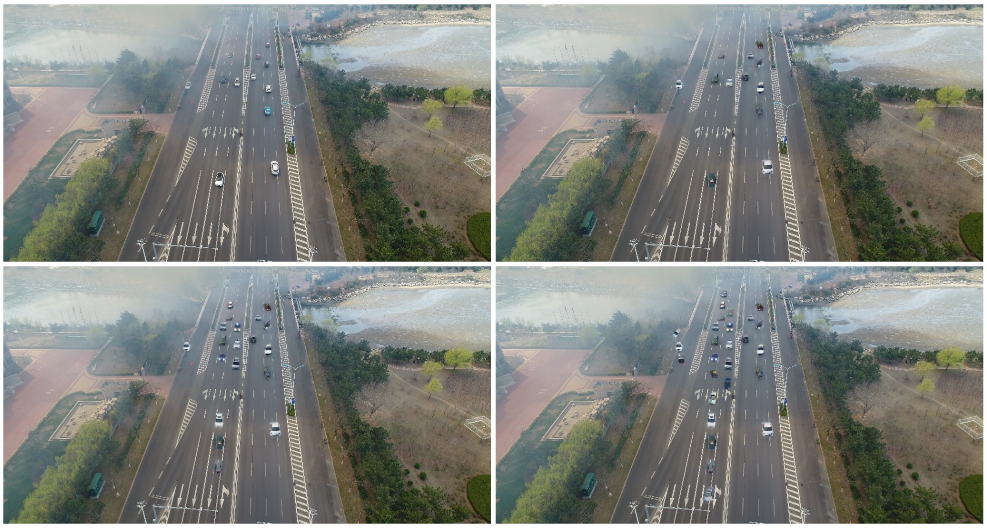


Figure 4.16: Data augmentation for small objects examples from UAVDT training set provided by our pipeline. From left to right and from top to bottom: standard real frame with LR objects; LR objects replaced by SLR objects; SLR×2 objects; and SLR×3 objects.

4.5 Conclusions

We have designed a novel a pipeline for data augmentation for small object detection, i.e., objects under 32×32 pixels. The pipeline takes a video dataset as input and returns the same dataset but with the images populated with annotated small synthetic objects. Thereby, by increasing the number of small objects —which are scarce in most of the datasets— the performance of detection models increases too. The main component of the pipeline is the small objects generation process. For this specific purpose we have designed Downsampling GAN (DS-GAN), a generative adversarial network based on the latest super-resolution techniques to generate high quality small objects taking large objects as a starting point —which are normally available in a high quantity.

The quality of small objects generated by DS-GAN has been validated in an isolated way. Experiments show that the FID value for the SLR objects is very close to the FID value for real LR objects, as opposed to the simple downsampled objects, which have a very distant FID value. In addition, we reached the same conclusion by training a standard CNN classifier. So that, we confirm that small objects generated by DS-GAN boost small object classification.

On the contrary, the small objects generated by direct large objects re-scaling are useless for data augmentation to recognise small objects, as the artefacts introduced by these functions differ greatly from real-world small objects.

The proposed pipeline for data augmentation method improves the performance of state-of-the-art models in the detection of small objects over the UAVDT dataset. The results over the test set demonstrate an improvement of 4.9% $\mathrm{AP}_s^{@[.5,.95]}$ and 11.9% $\mathrm{AP}_s^{@.5}$ for FPN [78] and of 4.4% $\mathrm{AP}_s^{@[.5,.95]}$ and 10.3% $\mathrm{AP}_s^{@.5}$ for STDnet [9] in a scenario where the number of training small objects is limited –only 25% of the videos are considered. These results validate the initial hypothesis that, when a dataset contains few small objects, the proposed data augmentation technique boosts the performance of the detector.

As future work, we plan to enhance the pipeline by bringing time consistency between synthetic objects within the same video, and also, by integrating the segmentation stage in DS-GAN to compute more accurate small object masks.

Chapter 5

Conclusions

Small object detection has become increasingly relevant due to the fact that the performance of common object detectors falls significantly as objects become smaller. Many computer vision applications require the analysis of the entire set of objects in the image, including extremely small objects. Moreover, the detection of small objects allows to perceive objects at a greater distance, thus giving more time to adapt to any situation or unforeseen event.

In this PhD Thesis the topic of small object detection has been addressed through deep learning techniques. Particularly, the work has focused on designing CNN-based architectures able to detect extremely small objects —under 16×16 pixels—, in both still images and videos through spatio-temporal processing. This work is complemented with the development of a system aimed to automatically increase the number of instances of small objects in a given dataset based on a generative adversarial network (GAN) approach for data augmentation.

In the field of image object detection, we have proposed a new architecture, called Small Target Detection network (STDnet), a region-proposal-based ConvNet to detect extremely small objects. The key of STDnet is the Region Context Network (RCN) and the RoI Collection Layer (RCL), an additional visual attention mechanism that allows STDnet to focus only on promising regions. This module prevents the processing of more than the 85% of the input image from this point onward. This saving in computational cost is employed to preserve fine-grained feature maps throughout the model, which is critical to ensure that small objects are represented with high-semantics in deep layers. This architecture leads to greater precision for detecting extremely small objects without increasing the computational cost and achieving state-of-the-art results in USC-GRAD-STDdb and the well-known MS COCO, UAVDT and

VisDrone datasets.

Then, in terms of video object detection, STDnet-ST has been introduced, a novel spatio-temporal ConvNet to detect extremely small objects in video. STDnet-ST is composed of two STDnet branches, and it binds the most likely candidate regions of two input frames by a correlation module to seek temporal coherence. This correlation-based approach outperforms common overlapping assessments as they do not work properly when dealing with small or fast objects. Moreover, since the correlation module does not rely on the spatial position of the objects, it can identify the same small object in two non consecutive frames, which is useful in applications where the videos have a low frame rate. The detections enclosed in the correlated regions are gradually joined to temporal tubelets to be further refined by a tubelet linking stage. The tubelet linking applies a Viterbi algorithm extension to the detections based on correlation linking, and implements a tubelet suppression procedure that allows STDnet-ST to dismiss unprofitable tubelets while preserving only high-quality ones. STDnet-ST improves the state-of-the-art results in USC-GRAD-STDdb, UAVDT and VisDrone video datasets by at least 2% $AP_{xs}^{@.5}$.

Finally, this thesis has contributed to address some drawbacks in the fact that object detection models are biased towards larger objects due to the lack of datasets with a large number of small objects: (i) we have released the aforementioned USC-GRAD-STDdb, a video dataset focused on extremely small objects with more than 25,000 annotated frames in complex backgrounds with clutter; and (ii) a novel pipeline for data augmentation for small —less than 32×32 pixels— and extremely small —less than 16×16 pixels— object detection has been proposed. This pipeline populates a given video dataset with annotated small synthetic objects. Thereby, by adding different small objects, the performance of detection models increases too due to larger training sets. The main component of the pipeline is the Downsampling GAN (DS-GAN), a generative adversarial network based on the latest super-resolution techniques to generate high quality small objects taking large objects as a starting point —which are normally available in a high quantity. In addition to DS-GAN, the pipeline consists of several computer vision processes: object segmentation, object inpainting and object blending. The combination of these processes enables the data augmentation system to generate high-quality small synthetic objects, to compute a coherent position inside a new frame, and to insert the object in accordance with the background. The proposed data augmentation algorithm was evaluated on the UAVDT video dataset, where all the studied ConvNets performed better trained with the augmented synthetic objects than without them. The use of augmented data boosted the performance in more than 10% $AP_s^{@.5}$ for a scenario with 25% of the training videos, i.e., with a small amount of training data.

As a result of the work developed in this thesis, some directions of research that might be worthy to explore in the future have been identified. These ideas focus on further improving the efficiency of the proposed object detector architectures and data augmentation system.

One promising research direction is to adapt a top-down pathway similar to that of the FPN into the STDnet architecture. The objective is to use Region Context Network (RCN) in early stages of the neural network, as has been done so far, but without keeping high-resolution feature maps, i.e., using the standard stride. As by using low-resolution feature maps in deep layers, the performance in detecting small objects is reduced, the plan is to apply a top-down pathway in which the RCN regions would be concatenated at various resolutions. This will lead to high-resolution feature maps at early stages with high semantics. So that, the whole computational benefit obtained by STDnet with RCN (over 85%) would not be diminished in deeper layers.

When it comes to spatio-temporal object detection, an interesting direction is to exploit several frames simultaneously in the deep feature aggregation. The idea is to forward several frames at the same time and aggregate the RoI pooling features so that the architecture itself is capable of generating large spatial-temporal tubelets. The current correlation map will be concatenated with the aggregated features to be more precise and the tubelet suppression algorithm will perform as before, but on the tubelets generated by the aggregated features.

Finally, future research in data augmentation for small object detection will include enhancing the pipeline by bringing time consistency between synthetic objects within the same video, and also, by integrating the segmentation stage in the Downsamplig GAN (DS-GAN) to compute more accurate small object masks. The former will be addressed by using the computed camera motion by perspective transformation in the position selector procedure, and the already known object motion, to evaluate if it is coherent a specific place for the same objects in subsequent frames. For the latter, the DS-GAN architecture will be modified to be able to generate small objects with their own masks. To do this, the input must be objects with a pre-computed mask channel.

Bibliography

[1] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

[2] Yancheng Bai, Yongqiang Zhang, Mingli Ding, and Bernard Ghanem. Finding tiny faces in the wild with generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21–30, 2018.

[3] Connelly Barnes, Eli Shechtman, Dan B Goldman, and Adam Finkelstein. The generalized patchmatch correspondence algorithm. In *European Conference on Computer Vision (ECCV)*, pages 29–43, 2010.

[4] Sean Bell, C. Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2874–2883, 2016.

[5] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *ACM SIGGRAPH Computer Graphics*, pages 417–424, 2000.

[6] Gedas Bertasius, Lorenzo Torresani, and Jianbo Shi. Object detection in video with spatiotemporal sampling networks. In *European Conference on Computer Vision (ECCV)*, pages 331–346, 2018.

[7] N. K. Bose. Superresolution from image sequence. In *Applied Imagery Pattern Recognition (AIPR) Workshop*, pages 81–86, 2003.

[8] Brais Bosquet, Manuel Mucientes, and Víctor M. Brea. STDnet: A ConvNet for Small Target Detection. In *British Machine Vision Conference (BMVC)*, 2018.

[9] Brais Bosquet, Manuel Mucientes, and Víctor M. Brea. STDnet: Exploiting high resolution feature maps for small object detection. *Engineering Applications of Artificial Intelligence*, 91:103615, 2020.

[10] Adrian Bulat, Jing Yang, and Georgios Tzimiropoulos. To learn image super-resolution, use a gan to learn how to do image degradation first. In *European Conference on Computer Vision (ECCV)*, pages 185–200, 2018.

[11] Peter J. Burt and Edward H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, 1983.

[12] Zhaowei Cai, Quanfu Fan, Rogerio Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision (ECCV)*, pages 354–370, 2016.

[13] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6154–6162, 2018.

[14] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017.

[15] Changrui Chen, Yu Zhang, Qingxuan Lv, Shuo Wei, Xiaorui Wang, Xin Sun, and Junyu Dong. RRNet: A hybrid detector for object detection in drone-captured images. In *IEEE International Conference on Computer Vision (ICCV) Workshops*, 2019.

[16] Yihong Chen, Yue Cao, Han Hu, and Liwei Wang. Memory enhanced global-local aggregation for video object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10337–10346, 2020.

[17] Ondrej Chum and Andrew Zisserman. An exemplar model for learning object classes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[18] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3642–3649, 2012.

[19] Daniel Cores, Manuel Mucientes, and Víctor M. Brea. RoI feature propagation for video object detection. In *European Conference on Artificial Intelligence (ECAI)*, pages 2680–2687, 2020.

[20] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 379–387, 2016.

[21] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893, 2005.

[22] Nasser Dardas, Qing Chen, Nicolas D. Georganas, and Emil M. Petriu. Hand gesture recognition using bag-of-features and multi-class support vector machine. In *IEEE International Symposium on Haptic Audio Visual Environments and Games*, pages 1–5, 2010.

[23] Jiajun Deng, Yingwei Pan, Ting Yao, Wengang Zhou, Houqiang Li, and Tao Mei. Relation distillation networks for video object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 7023–7032, 2019.

[24] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012.

[25] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision (ECCV)*, pages 184–199, 2014.

[26] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2015.

[27] Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, and Qi Tian. The unmanned aerial vehicle benchmark: Object detection and tracking. In *European Conference on Computer Vision (ECCV)*, pages 370–386, 2018.

[28] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1033–1038, 1999.

[29] Christian Eggert, Dan Zecha, Stephan Brehm, and Rainer Lienhart. Improving small object proposals for company logo detection. In *ACM International Conference on Multimedia Retrieval (ICMR)*, pages 167–174, 2017.

[30] Mark Everingham, Luc Van Gool, Christopher Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[31] Sina Farsiu, M. Dirk Robinson, Michael Elad, and Peyman Milanfar. Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing*, 13(10):1327–1344, 2004.

[32] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3038–3046, 2017.

[33] Pedro Felzenszwalb, Ross Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[34] Mauro Fernández-Sanjurjo, Brais Bosquet, Manuel Mucientes, and Víctor M. Brea. Real-time visual detection and tracking system for traffic monitoring. *Engineering Applications of Artificial Intelligence*, 85:410–420, 2019.

[35] Dustin Franklin. NVIDIA Jetson TX2 delivers twice the intelligence to the edge. *NVIDIA Accelerated Computing Parallel Forall*, 2017.

[36] Gilad Freedman and Raanan Fattal. Image and video upscaling from local self-examples. *ACM Transactions on Graphics*, 30(2):1–11, 2011.

[37] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander Berg. DSSD: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[38] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1134–1142, 2015.

[39] Ross Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[40] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[41] Georgia Gkioxari and Jitendra Malik. Finding action tubes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 759–768, 2015.

[42] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

[43] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[44] Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, and R. Venkatesh Babu. DeLiGAN: Generative adversarial networks for diverse and limited data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 166–174, 2017.

[45] Bharath Hariharan, Pablo Arbelaez, Ross Girshick, and Jitendra Malik. Object instance segmentation and fine-grained localization using hypercolumns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):627–639, 2017.

[46] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.

[47] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[48] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pages 630–645, 2016.

[49] Li He, Hairong Qi, and Russell Zaretzki. Beta process joint dictionary learning for coupled feature spaces with application to single image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 345–352, 2013.

[50] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6626–6637, 2017.

[51] Sanghoon Hong, Byungseok Roh, Kye-Hyeon Kim, Yeongjae Cheon, and Minje Park. PVANET: Deep but lightweight neural networks for real-time object detection. *arXiv preprint arXiv:1611.08588*, 2016.

[52] Peiyun Hu and Deva Ramanan. Finding tiny faces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 951–959, 2017.

[53] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017.

[54] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[55] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics*, 36(4):1–14, 2017.

[56] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[57] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1125–1134, 2017.

[58] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM International Conference on Multimedia*, pages 675–678, 2014.

[59] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, pages 694–711, 2016.

[60] Yannis Kalantidis, Lluis Pueyo, Michele Trevisiol, Roelof van Zwol, and Yannis Avrithis. Scalable triangulation-based logo recognition. In *ACM International Conference on Multimedia Retrieval (ICMR)*, page 20, 2011.

[61] Vicky Kalogeiton, Vittorio Ferrari, and Cordelia Schmid. Analysing domain shift factors between videos and images for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2327–2334, 2016.

[62] Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, et al. T-CNN: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2896–2907, 2017.

[63] Kai Kang, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Object detection from video tubelets with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 817–825, 2016.

[64] Ramesh Kestur, Avadesh Meduri, and Omkar Narasipura. Mangonet: A deep semantic segmentation architecture for a method to detect and count mangoes in an open orchard. *Engineering Applications of Artificial Intelligence*, 77:59–69, 2019.

[65] Hyun I. Kim, Sung Shin, Wei Wang, and Soon I. Jeon. SVM-based Harris corner detection for breast mammogram image normal/abnormal classification. In *Research in Adaptive and Convergent Systems*, pages 187–191, 2013.

[66] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1637–1645, 2016.

[67] Kwang In Kim and Younghee Kwon. Single-image super-resolution using sparse regression and natural image prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1127–1133, 2010.

[68] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[69] Mate Kisantal, Zbigniew Wojna, Jakub Murawski, Jacek Naruniec, and Kyunghyun Cho. Augmentation for small object detection. *arXiv preprint arXiv:1902.07296*, 2019.

[70] Rolf Köhler, Christian Schuler, Bernhard Schölkopf, and Stefan Harmeling. Mask-specific inpainting with deep neural networks. In *German Conference on Pattern Recognition (GCPR)*, pages 523–534, 2014.

[71] Tao Kong, Anbang Yao, Yurong Chen, and Fuchun Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 845–853, 2016.

[72] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[73] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[74] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *IEEE*, pages 2278–2324, 1998.

[75] Yann LeCun, Corinna Cortes, and C. J. Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[76] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4681–4690, 2017.

[77] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Perceptual generative adversarial networks for small object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[78] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125, 2017.

[79] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[80] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[81] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.

[82] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*, 2018.

[83] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.

[84] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4898–4906, 2016.

[85] Wenchi Ma, Yuanwei Wu, Feng Cen, and Guanghui Wang. MDFN: Multi-scale deep feature learning network for object detection. *Pattern Recognition*, 100:107–149, 2020.

[86] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[87] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[88] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016.

[89] Thomas Moranduzzo and Farid Melgani. A SIFT-SVM method for detecting cars in UAV images. In *IEEE International Geoscience and Remote Sensing Symposium*, pages 6868–6871, 2012.

[90] MS COCO Leaderboard. Microsoft COCO detection leaderboard. `http://cocodataset.org/#detection-leaderboard`, 2019. Accessed: 2019-02-08.

[91] Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

[92] Kamal Nasrollahi and Thomas B. Moeslund. Super-resolution: a comprehensive survey. *Machine Vision and Applications*, 25(6):1423–1468, 2014.

[93] Junhyug Noh, Wonho Bae, Wonhee Lee, Jinhwan Seo, and Gunhee Kim. Better to Follow, Follow to Be Better: Towards precise supervision of feature super-resolution for small object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 9725–9734, 2019.

[94] Shuchao Pang, Juan José del Coz, Zhezhou Yu, Oscar Luaces, and Jorge Díez. Deep learning to frame objects for visual target tracking. *Engineering Applications of Artificial Intelligence*, 65:406–420, 2017.

[95] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, 2000.

[96] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544, 2016.

[97] Xiaojiang Peng and Cordelia Schmid. Multi-region two-stream R-CNN for action detection. In *European Conference on Computer Vision (ECCV)*, pages 744–759, 2016.

[98] Xiaojiang Peng and Cordelia Schmid. Multi-region two-stream r-cnn for action detection. In *European Conference on Computer Vision (ECCV)*, pages 744–759, 2016.

[99] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM SIGGRAPH Transactions on Graphics*, 22(3):313–318, 2003.

[100] Thomas Porter and Tom Duff. Compositing digital images. In *ACM SIGGRAPH Computer Graphics*, pages 253–259, 1984.

[101] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016.

[102] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[103] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[104] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[105] Jimmy S. J. Ren, Li Xu, Qiong Yan, and Wenxiu Sun. Shepard convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 901–909, 2015.

[106] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[107] Stefan Romberg, Lluis Pueyo, Rainer Lienhart, and Roelof van Zwol. Scalable logo recognition in real-world images. In *ACM International Conference on Multimedia Retrieval (ICMR)*, pages 25:1–25:8, 2011.

[108] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision (ECCV)*, pages 430–443, 2006.

[109] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. Detecting flying objects using a single moving camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(5):879–892, 2017.

[110] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[111] Ikuro Sato, Hiroki Nishimura, and Kensuke Yokoi. APAC: Augmented pattern classification with neural networks. *arXiv preprint arXiv:1505.03229*, 2015.

[112] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[113] Assaf Shocher, Nadav Cohen, and Michal Irani. “zero-shot” super-resolution using deep internal learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3118–3126, 2018.

[114] Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. Summarizing visual data using bidirectional similarity. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[115] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NIPS)*, pages 568–576, 2014.

[116] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[117] Jian Sun, Zongben Xu, and Heung-Yeung Shum. Image super-resolution using gradient profile prior. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[118] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[119] Richard Szeliski, Matthew Uyttendaele, and Drew Steedly. Fast poisson blending using multi-splines. In *IEEE International Conference on Computational Photography (ICCP)*, 2011.

[120] Masayuki Tanaka, Ryo Kamio, and Masatoshi Okutomi. Seamless image cloning by a closed form solution of a modified poisson problem. In *SIGGRAPH Asia 2012 Posters*. ACM, 2012.

[121] Peng Tang, Chunyu Wang, Xinggang Wang, Wenyu Liu, Wenjun Zeng, and Jingdong Wang. Object detection in videos by high quality object linking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[122] Xiaoyu Tao, Yihong Gong, Weiwei Shi, and De Cheng. Object detection with class aware region proposal network and focused attention objective. *Pattern Recognition Letters*, 130:353–361, 2020.

[123] Tong Tong, Gen Li, Xiejie Liu, and Qinquan Gao. Image super-resolution using dense skip connections. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4799–4807, 2017.

[124] Ken Turkowski. Filters for common resampling tasks. In *Graphics Gems*, pages 147–165, 1990.

[125] Jasper R. R. Uijlings, Koen E. A. Van De Sande, Theo Gevers, and Arnold W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[126] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[127] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning (ICML)*, pages 1058–1066, 2013.

[128] Juan Wang, Xiaoming Tao, Mai Xu, Yiping Duan, and Jianhua Lu. Hierarchical objectness network for region proposal generation and object detection. *Pattern Recognition*, 83:260–272, 2018.

[129] Shiyao Wang, Yucong Zhou, Junjie Yan, and Zhidong Deng. Fully motion-aware network for video object detection. In *European Conference on Computer Vision (ECCV)*, pages 542–557, 2018.

[130] Xiaolong Wang, Abhinav Shrivastava, and Abhinav Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2606–2615, 2017.

[131] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. ESRGAN: Enhanced super-resolution generative adversarial networks. In *European Conference on Computer Vision (ECCV) Workshops*, 2018.

[132] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding*, 193:102907, 2020.

[133] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):463–476, 2007.

[134] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. GP-GAN: Towards realistic high-resolution image blending. In *ACM International Conference on Multimedia*, pages 2487–2495, 2019.

[135] Fanyi Xiao and Yong Jae Lee. Video object detection with an aligned spatial-temporal memory. In *European Conference on Computer Vision (ECCV)*, pages 485–501, 2018.

[136] Zhiwei Xiong, Xiaoyan Sun, and Feng Wu. Robust web image/video super-resolution. *IEEE Transactions on Image Processing*, 19(8):2017–2028, 2010.

[137] Bin Yang, Junjie Yan, Zhen Lei, and Stan Z. Li. Craft objects from images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6043–6051, 2016.

[138] Chih-Yuan Yang, Chao Ma, and Ming-Hsuan Yang. Single-image super-resolution: A benchmark. In *European Conference on Computer Vision (ECCV)*, pages 372–386, 2014.

[139] Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2129–2137, 2016.

[140] Jianchao Yang, Zhe Lin, and Scott Cohen. Fast image super-resolution based on in-place example regression. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1059–1066, 2013.

[141] Jianchao Yang, John Wright, Thomas Huang, and Yi Ma. Image super-resolution as sparse representation of raw image patches. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[142] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. WIDER FACE: A face detection benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5525–5533, 2016.

[143] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Generative image inpainting with contextual attention. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5505–5514, 2018.

[144] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Free-form image inpainting with gated convolution. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4471–4480, 2019.

[145] Jin Yuan, Heng-Chang Xiong, Yi Xiao, Weili Guan, Meng Wang, Richang Hong, and Zhi-Yong Li. Gated CNN: Integrating multi-scale feature layers for object detection. *Pattern Recognition*, pages 107–131, 2019.

[146] S. Zagoruyko, A. Lerer, T.-Y. Lin, P. O. Pinheiro, S. Gross, S. Chintala, and P. Dollár. A multipath network for object detection. In *British Machine Vision Conference (BMVC)*, 2016.

[147] Dan Zeng, Fan Zhao, Shiming Ge, and Wei Shen. Fast cascade face detection with pyramid network. *Pattern Recognition Letters*, 119:180–186, 2018.

[148] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z. Li. S3fd: Single shot scale-invariant face detector. In *IEEE International Conference on Computer Vision (ICCV)*, pages 192–201, 2017.

[149] Yongqiang Zhang, Mingli Ding, Yancheng Bai, and Bernard Ghanem. Detecting small faces in the wild based on generative adversarial network and contextual information. *Pattern Recognition*, 94:74–86, 2019.

[150] Qiaoyong Zhong, Chao Li, Yingying Zhang, Di Xie, Shicai Yang, and Shiliang Pu. Cascade region proposal and global context for deep object detection. *Neurocomputing*, 2019.

[151] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Conference on Artificial Intelligence (AAAI)*, 2020.

[152] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2223–2232, 2017.

[153] Pengfei Zhu et al. VisDrone-VID2019: The vision meets drone object detection in video challenge results. In *IEEE International Conference on Computer Vision (ICCV) Workshops*, 2019.

[154] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 408–417, 2017.

[155] Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. Traffic-sign detection and classification in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2110–2118, 2016.

[156] Barret Zoph, Ekin D. Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V. Le. Learning data augmentation strategies for object detection. *arXiv preprint arXiv:1906.11172*, 2019.

List of Figures

List of Tables