UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech

UNIVERSITAT DE BARCELONA (UB)

UNIVERSITAT ROVIRA i VIRGILI (URV)

Master in Artificial Intelligence

Recovering missing data from human body images

Martí Soler Planas
Advisor: Sergio Escalera - Dpt. de Matemàtiques i Informàtica UB

January 2018

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

FACULTAT DE MATEMÀTIQUES (UB)

ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA (URV)

Contents

Abstract

In recent years advancements in generative models and deep unsupervised learning have allowed a wide variety of methods to thrive on problems traditionally considered too challenging for computers this has been achieved thanks to the capability of new models as well as the current availability of big amounts of data. One of these problems is the current hot topic of semantic image inpainting. In this work we engage in a critical discussion about the state of the art in semantic inpainting and related fields, as well as compare some of the most prominent methods. We think this analysis is important due to the novelty of both generative and deep unsupervised techniques which are an integral part of these methods. Solutions to this problem usually require a large amount of semantic information to generate a proper image reconstruction. To mitigate that, we restrict the domain of the problem to semantic inpainting of still images containing humans by creating a new specialized dataset. To qualify the performance of these methods we use traditional metrics such as MSE and DSSIM, combined with a novel metric based on quality of pose evaluation techniques over the reconstruction. We argue that there are some very prominent problems on the field which there is few discussion about and no clear consensus.

Keywords. Computer Vision, Semantic Inpainting, Unsupervised Learning, Human Pose Analysis, Generative Models, Deep learning

1 Introduction

In recent years computers have been getting much better at tasks requiring deep understanding of images. In even more recent years it has been shown that they can use this learned knowledge to be able to generate new images. Adding these two premises together follows as the next logical step. There has been an increase of models capable of not only understanding the image, but also what is missing from it and how to blend this new information seamlessly.
The task just defined is called semantic inpainting and, as its name implies, consists of reconstructing missing parts of the image. A requirement for the task to be considered **semantic** inpainting is that these parts must be big enough that they cannot be reconstructed with traditional interpolation techniques and require semantic knowledge.
In this work we will start by defining the problem in more detail, together with its complexities, and a brief introduction to the state of the art. Next, we will collect a dataset containing images of humans to see how body parts and backgrounds can be reconstructed to help on the task of human pose estimation. We will process this dataset to adjust it to this problem. To generate this dataset, we will select already existing widely used datasets and filter, modify and normalize them so that they can be merged into a dataset that fits our needs.
After that, we will pick some of the current state of the art methods for semantic inpainting and provide a deep analysis of how and why they work, as well as how they tackled the problem. We will define some evaluation criteria, consisting of traditional measures together with a new measure contributed by us. We think a combination of these measures will ensure that we capture all the intricacies of the results.
We proceed to analyze and optimize the parameters of the selected methods, this is to ensure both, that our understanding about the methods is correct and that they have good performing parameters for our dataset.
Then we will proceed to a deep discussion about the state of the art, focused mainly on the methods used, remarking common patterns from these methods and the problems they aim to solve.
We will also present a competition challenge prepared alongside this work to allow people to use the same dataset prepared earlier, having as baselines the methods analyzed and which we hope ends up pushing the state of the art in this field forward. Finally, we will state the final conclusions about this work as well as some possible further work.

1.1 Problem

We focus on the concrete problem of inpainting images of humans, that is recovering RGB pixel level information from missing regions of still images containing humans.
Given an image masked by multiple blocks and the position and size of these blocks, we want to be able to restore the masked parts of the image in a way that resembles the original content and looks plausible to a human. In order

to evaluate the recovery of the missing parts, we will evaluate common used metrics such as MSE and DSSIM, together with human pose estimation score.
The masks will have large zones of contiguous masked area, and this causes some additional challenges. Reconstructions tend to be easier near real points of data, as this allows the model to interpolate between them, but in our case we are dealing with masks big enough where the onion problem comes into play. The first pixels near the border of the mask are somewhat easy to predict as we are near real data, but for the next row of pixels, we will have to extrapolate from the previous row, which was already extrapolated data. And this continues iteratively until the center of the mask. This iteration process causes the error generated in each step to add up extremely fast.
Another problem we have with such a big mask is that we are no longer predicting how a structured pattern will continue. If the mask occludes a hand, the model needs to have knowledge that the thing above the hand is an arm, that a hand is usually attached to the arm, and the information itself about how a hand looks. This imposes the requirement that models tackling this task must have the ability to learn and quickly interpret a vast amount of semantic information.

1.2 Motivation

This is a current hot topic due to advancements in generative models, but from our research, other works expect their model to learn semantic information about a wide array of different topics. We believe that restricting the domain to images depicting humans can greatly diminish the amount of semantic information the problem requires, thus allowing far better results.
Solutions to this problem may be useful in a wide variety of tasks, for example data augmentation, films or as a preprocessing technique to allow the use of pose recognition algorithms on images containing occlusions.
It is not a stretch to imagine that future self driving cars might use pose recognition techniques to guess whether a pedestrian is about to step into the road. In this case, pedestrians being partially occluded by an object such as a traffic signal may become a common occurrence. In this case, we could use inpainting as a preprocess before pose estimation techniques to increase the accuracy of the latter.

1.3 Related Work

Traditionally, there have been a lot of methods proposed to recover masked information on images. The first batches of more successful methods thrived at reconstructing a texture or a repeating pattern either by using the structure of the textures [6, 5, 10] or by using patches [13], this sub-problem was sometimes called image quilting.
Recent advances in computer vision, partially led by Convolutional Neural Networks (CNNs) [12] have shown that these models are capable of learning a great amount of semantic information, useful in tasks such as classification or segmentation. There have also been advances in reconstructing missing infor-

mation by using models such as denoising autoencoders [20], which did so via a model capable of extracting relevant features and another model capable of reconstructing information from these features.
Even more recently there have been big advances in generative models with Generative Adversarial Networks (GANs) [9]. For the problem of semantic inpainting this does not only provide a good generative model, but also a way to qualify some qualities of the results that traditional measures were not able to measure accurately.

2 Dataset

The proposed dataset is composed of **41076** images, where each image is centered on a human which may or may not be partially occluded. The ratio of background information in relation to human body is roughly the same for all images although some other humans may appear in the background with an arbitrary size. The humans appear in a lot of poses with a variety of difficulties from simple rigid poses to complicated sport-oriented poses containing multiple self occlusions. They were also manually labeled with joint information about the targeted human. We split the data into **60%** training, **20%** Validation and **20%** test. We also generated masks occluding parts of the image for each set.
In fig. 1 we can see a visual example of how applying inpainting techniques can improve the results of a state of the art pose estimation technique.

2.1 Objectives of the dataset

While searching for data to use in our dataset, we wanted to define some criteria by which to filter datasets which were not fit for our needs. So, we established the following requirements:

- **Humans:** As we are working in a problem of analyzing humans, we required that all the datasets depicted humans in a prominent position. In one of the processing steps we extract crops of the original which contain each human in a central position, so we heavily benefit from images with multiple humans.
  We did add the requirement that the humans were not too far away from the camera or completely occluded.

- **Resolution:** We required our dataset to have proper resolution. The original image resolution was not important to us, as our aim was that humans on the photo appeared bigger than 64x64px, so that was the requirement added. We chose this resolution because it is commonly seen in other methods.

- **Variance:** We wanted our dataset to have high variance, so we tried to either include datasets with a high variance themselves or that the ones with low variance were not a big part of the final dataset.

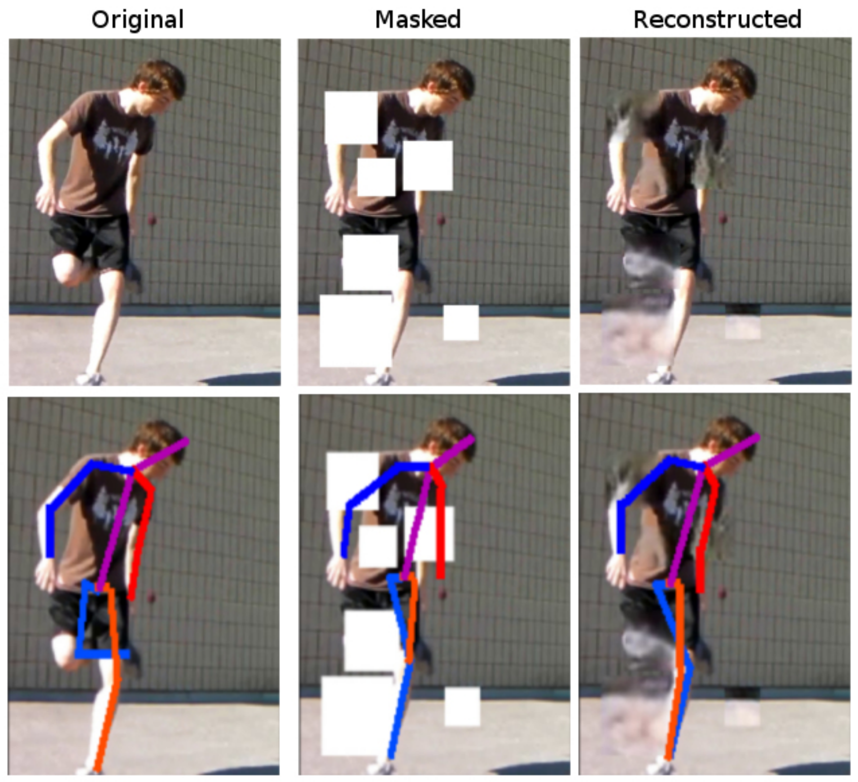


Figure 1: Visual comparison of pose estimation results on the same image. This example has been cherrypicked to show a place where the reconstruction improves the pose estimation (left arm) and a place where the reconstruction is not good enough (right leg).

- **Pose:** For one of our measures we will apply pose estimation methods, so we wanted our dataset to have manually labeled poses. We did restrict datasets depending on the criteria used for labeling, to avoid having labels which could not be standardized later.
- **Variance of poses:** To really prove that the models are capable of learning semantic information about poses, we wanted to have a wide range of poses in both, variance and difficulty. To accomplish this, we did not use video frames which were temporally close together or datasets with low pose variance.

2.2 Sources

We collected images from multiple sources which fit the criteria detailed in section 2.1.
The final sources used can be seen in fig. 2.2. They have been selected because they provide a high variability of content while being representative enough to allow the resulting methods to generalize.

Name	**#Images Used**	**Cropped**
MPII Human Pose Dataset [Andriluka et al. (2014)]	26571	Yes
Leeds Sports Pose Dataset [Johnson and Everingham (2010)]	2000	No
Synchronic Activities Stickmen V [Eichner and Ferrari (2012)]	1128	Yes
Short BBC Pose [Charles et al. (2013)]	996	No
Frames Labelled In Cinema [Sapp and Taskar (2013)]	10381	Yes

Table 1: Source comparison. Note that the number of images used is not the same as the original size of the dataset. This is due to extracting multiple human centered crops from each image and filtering out the ones which did not fulfill some of the requirements.

2.3 Pre-process

In this section we explain the multiple pre-processing steps applied to the images of the dataset.

2.3.1 Cropping

Most of the original images contained big scenes with one or more humans, so we generated multiple crops. A crop is created for every human with joint information contained in the dataset, except for the ones which were either too small or only contained one annotated joint.
The only datasets which we did not have to crop were the Leeds Sports Pose Database and the Short BBC Pose as it can be seen in fig. 2.2. For the others, we applied a standard procedure consisting in calculating the minimum bounding box containing all joints, and then enlarging this box enough to fit some of the context of the image, while keeping a similar aspect ratio.
For the FLIC dataset which came from videos we decided on skipping some of the frames to increase the pose variance. It is not possible for us to give an accurate measure of these skips as the framerate of the original is not regular. In the end, we took around half of the original frames, leaving us with a maximum theorical framerate of 5 frames per second, but which from our qualitative analysis is much closer to 1 frame every 3 seconds.

2.3.2 Labeling

To make the joint labels consistent we made sure that all the selected datasets used the same criteria when annotating each body part. It is important to note that while not every image has all joints labeled, the set of annotations for each body part is consistent. In fig. 2 we can see which is the criteria for labeling each joint we consider as valid.
In fig. 2.3.2 we can see which joints we extracted from each dataset. The only ones that appear throughout all the datasets are **wrists**, **elbow** and **shoulder**.

Name	**#Joints**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
MPII	19	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
LSP	8	✓	✓	✓			✓	✓	✓	✓	✓		
Synch	6					✓	✓	✓	✓	✓	✓		
BBC	4							✓	✓	✓	✓		
FLIC	6			✓					✓	✓	✓	✓	✓

Table 2: Checkmarks indicate that either the joint appears in some images of the dataset or its position can be inferred from the original annotations. Joints refer to 1-**Ankles**, 2-**Knees**, 3-**Hips**, 4-**Pelvis**, 5- **Thorax**, 6-**Upper Neck**, 7-**Head Top**, 8-**Wrists**, 9-**Elbows**, 10-**Shoulders**, 11-**Eyes**, 12-**Nose**

Some adjustments had to be done to ensure the annotations are consistent, these are:

- We removed the information about right and left of the joints since we had both, some datasets used the camera as a reference and others used the human. It was not possible to standardize to the same reference as we had multiple datasets with humans facing backwards.

- In the Synchronic dataset we did not have a thorax joint, but it contained a vertical line through the upper body and the middle of that line was at the thorax position, so we decided to use that as our label for the thorax instead.

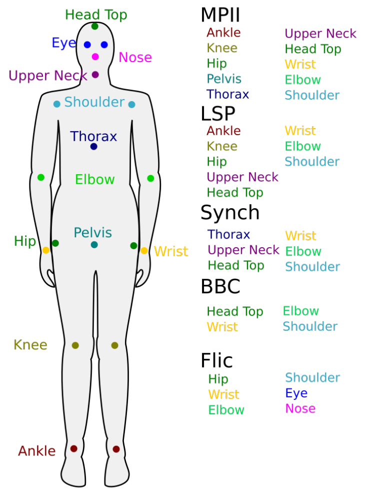


Figure 2: Visualization of the selected joints and their position.

2.3.3 Masking

For each image we generated a different mask to hide parts of the image. We will analyze how well algorithms will be able to restore the parts of the image occluded by this mask.

We set the following procedure: masks will consist of N blocks, where $0 < N < 11$ for each image, being each block a square of size s ranging from

$$\frac{min(w,h)}{20} < s < \frac{min(w,h)}{3}$$

where w and h are the width and height of the image respectively. These block positions will be mostly random, but we will introduce a bias pushing them towards covering joints. There will be no overlap between blocks and they will always have a margin of 100px from the image's edge. At most 70% of the image will be masked.
In fig. 3 we can appreciate that most of the blocks have a size between 20px and 80px, but we also have a significant amount of big blocks up to 170px. These big blocks may serve to penalize models that base most of their reconstructions from pure context instead of using learned semantic knowledge conditioned by the context. From this figure we can also infer that most images do not have a lot of blocks position covering joints, but it is still worthwhile to make a system that learns human parts because most of the blocks end up being on joints. We also have a gaussianlike distribution for the number of blocks in each image, where the left side has been slightly skewered due to not having images with negative amounts of blocks.

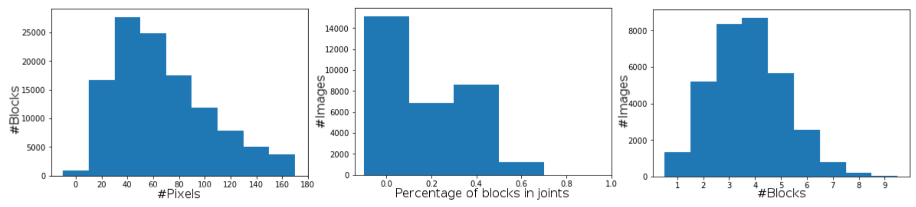


Figure 3: Histograms showing the distributions of different properties of the masks on the train and validation sets.

3 Analyzed Methods

In this section we will look at some of what we consider the most representative state of the art techniques to solve semantic image inpainting. None of the methods specified in this section is specialized in the human domain. We will provide an explanation and posterior analysis of these methods, we believe that even if they are not tailor-made for the human domain they can be trained to specialize in this problem.

- [Pathak et al. (2016)] This method uses an autoencoder model with an encoder which architecture is derived from AlexNet, connected to a decoder through a new type of layer. This autoencoder is later used as a generator together with a discriminator in an adversarial model. This model was

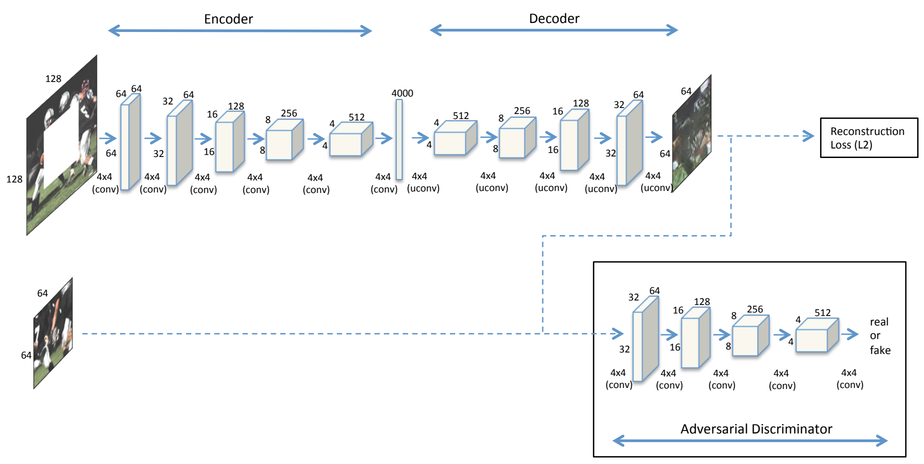


Figure 4: Architecture of ContextEncoders (source: [15]).

trained by a conjunction of a reconstruction loss (normalized masked L2) and an adversarial loss provided by a discriminator.

- [Yeh* et al. (2017)] Its main strength is the ability to capture high level context. The dataset originally used by this method consisted of faces, but we think that the method can be applied to this problem because capturing the details of the different distributions of shapes in the human body is an integral challenge to our problem. To do so, this method uses the Deep Convolutional model to learn a semantic space and then a new vector is iterated on to search for a good encoding in that latent image manifold.

- [Yang et al. (2017)] This baseline is centered in obtaining high resolution results. To do so, it uses a model based on two networks, a content network tasked with creating an initial approximation, and a texture network which is tasked with adding high frequency details by constraining the texture of the generated image according to the texture of the non masked area. Context Encoders (Our first baseline) are used as the content prediction network which serves as initialization for the multi-scale algorithm.

As we can see all of these methods use in one way or another Generative Adversarial networks, as they are one of the most prominent generative models. We will proceed to analyze these methods individually and in more detail.

3.1 Context Encoders

Context Encoders defined in [Pathak et al. (2016)] is a model heavily inspired by two common architectures, denoising autoencoders [19] and DCGAN [17].
Its architecture, which can be seen in fig. 4 consists of an encoder, a decoder and a discriminator. The encoder is tasked with learning a compact representation

of the image containing both, semantic and low level information, but as shown in [19] if left to its own device, the encoder would just learn how to compress the data, in [19] they avoid this by adding noise to the input, but in our case the distortion we want to reconstruct is big enough to push the model to learn semantic information. This model uses two joint losses, a basic measure of reconstruction quality (ℓ_2) and an adversarial loss provided by a jointly trained discriminator, which learns to classify the output of the generator, in this case the autoencoder, into real or fake images. This provides another measure of quality of the reconstruction that incentivizes choosing a particular mode of the distribution instead of the mean. This greatly improves one of the classic problems of parametric models for semantic inpainting, blurry images.
Another improvement proposed by this model is to join the encoder and the decoder using what they call a channel-wise fully connected layer. This layer manages to enable a way to propagate information from each two points of the image while keeping a reasonable amount of parameters. This is usually not needed in problems when we are reconstructing the same image, but it is absolutely necessary for the semantic inpainting problem.
Next, we will look at some of the insights which allow this architecture to train correctly and give good results.

3.1.1 Loss definition

One of the main ideas behind this paper is to capture both of the main opinions of what is considered a good reconstruction, either it resembles the original or it is sensible with the context of the image. To insert this idea into the model they train it with a mixture of two different losses.

One of them being L2:

Reconstruction loss. Normalized masked L2 distance.

$$\mathcal{L}_{rec}(x) = \left|\left|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\right|\right|_2^2$$

And the other one being the adversarial loss from the discriminator. In this case the authors found out that they obtained better results by not conditioning the discriminator to the context and only conditioning the generator. The same happened when we tried it with our dataset. They attribute this to the discriminator exploiting the perceptual discontinuity in the region joining the reconstruction and the original.

Adversarial Loss.

$$\mathcal{L}_{adv} = \max_D \mathbb{E}_{x \in X}[log(D(x)) + log(1 - D(G((1 - \hat{M} \odot x)))]$$

In which G is the generator and D the discriminator. To join them they use:

Joint Loss.

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv}$$

Where λ_{rec} and λ_{adv} are weights that follow $\lambda_{rec} + \lambda_{adv} = 1$. Both of these losses evaluate the quality of the reconstruction, but they prioritize different properties. The Reconstruction loss measures the distance to the original, this measure has been an standard of traditional computer vision techniques and it provides a lot of stability.
Adversarial loss acts more like a human would act, and it penalizes images which seem unnatural. This is highly desired as the Reconstruction loss does have a tendency to choose the mean of the distribution, causing blurry images.

3.2 Channel-wise fully-connected layer

One of the properties caused by the model chosen is that information has trouble propagating from different corners of the image. This is due to the model being fully convolutional, which implies that it does not have any fully connected layer. This is usually the case for these generators as it heavily facilitates the training by reducing the amount of parameters, but in this particular case they argue this is not adequate for the problem.
They argue that in this problem, information contained in one corner of the image may be needed to reconstruct another corner, so the information should be able to propagate between any two points of the image. The simpler way to do this would be to use a fully connected layer after the bottleneck, but due to the big bottleneck required for the problem, that would result in a model almost impossible to train. If the model were to have m features of size nxn the number of parameters of this layer would be:

$$m^2n^4$$

To solve this they introduce a new kind of layer they call Channel-wise fully-connected layer, which manages to propagate information from all points of the image while keeping the number of parameters tractable.
This layer works similarly to a fully connected layer, but instead of connections between different feature maps, it only has connections **within** the same feature map. This decreases the number of parameters to:

$$mn^4$$

Which is easier to train. In the model, they use this layer at the bottleneck between the encoder and the decoder and after it they add a convolutional layer with stride 1 to help propagate the information.

3.3 Overlap

When looking at results of inpainting methods dealing with big continuous masks, one of the best ways to find out which part of the image was reconstructed, is to

search for the border between the original and the reconstruction. The sudden difference in style is sometimes a strong indicator of where the reconstruction begins. If this difference is strong enough, it can even cause the model to not train correctly, as the discriminator may only look at the borders to detect whether the image is real or fake, causing the generator to collapse.
That is why special care has to be put into learning how to generate the borders of the reconstruction correctly. To do so, this model will generate an extra amount of pixels around the original mask, this will be called overlap, as illustrated in fig. 5.

When calculating the loss of the prediction over the overlap zone, they use a higher weight (10x) for the reconstruction loss. This is a way to focus a bigger part of the effort while training, into obtaining a good reconstruction for the border. This does not only improve the final results, but it also increases the stability as it lessens the possibility of the discriminator guessing correctly by only looking at the border.

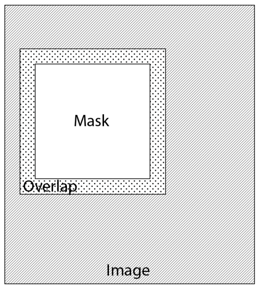


Figure 5: Depiction of how overlap is located around each block.

3.4 Semantic Image Inpainting with Deep Generative Models

The next model we will be looking at is [Yeh* et al. (2017)], this model is interesting because instead of training the model to reconstruct masked images it trains the model to represent the semantic space of these images. Then when a new image is given to the model it can move it around the semantic space until a reconstruction for the masked zone is found.
The main idea behind the paper is that all the encodings of unmasked images used for training will lay on a manifold, and any masked data given to the model will lay outside. If this is true, then they can recover the point in the manifold closest to the input data, and that will be the encoding of a valid reconstruction $\hat{\mathbf{z}}$.

$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z}}\{\mathcal{L}_c(\mathbf{z}|\mathbf{y}, \mathbf{M}) + \mathcal{L}_p(\mathbf{z})\},$$

Where M is the mask applied, y is the masked image, $\mathcal{L}_c$ is the context Loss and $\mathcal{L}_p$ is the prior Loss.
Once they obtain $\hat{\mathbf{z}}$ they can use their generator to generate the masked part of the image.

3.4.1 Loss definition

Another extremely important decision taken in this method is how to define the loss. Similarly to the case with Context Encoders, they decided on using a mixture of losses defined as:

- **Context Loss:** The idea for this loss is to capture the difference between the generated data and the real one. However, just applying the ℓ_2 norm would give equal importance over all the pixels. In the paper it is argued that pixels near an unmasked pixel should be easier to reconstruct and more important, thus they should have bigger weight. That's why they will apply a weighting term to each pixel dependant on how close to the mask it is.
$$\mathbf{W}_i = \begin{cases} \sum_{j \in N(i)} \frac{(1-\mathbf{M}_j)}{|N(i)|} & \text{if } \mathbf{M}_i \neq 0 \\ 0 & \text{if } \mathbf{M}_i = 0 \end{cases}$$
Where $\mathbf{W}_i$ is the weigh of the pixel i and $N(i)$ is the set of neighbours of pixel i considering a window of size 7. After this weighting mask is generated, they use the following formulation for the loss:
$$\mathcal{L}_c(\mathbf{z}|\mathbf{y}, \mathbf{M}) = \|\mathbf{W} \odot (G(\mathbf{z}) - \mathbf{y})\|_1.$$

- **Prior Loss:** This loss works much more like our intuition might expect, penalizing images which do not resemble the samples from the training set. To do this we will use the discriminator which was trained to differentiate different images.
$$\mathcal{L}_p(\mathbf{z}) = \lambda \log(1 - D(G(\mathbf{z}))).$$
Where λ is the weight of this loss, this helps offset some scaling problems between the losses.

3.4.2 Inference

Once the model is trained, using it for inference with new images is not as straighforward as in other models. As a result from the model being trained to generate a semantic space, they have to use some extra measures to inpaint the masked region. They use an iterative process which starts by feeding a random vector to the generator, and progressively adjust this vector via backpropagation to get a closer encoding to the input. This is generating the image on the semantic space closer to the input.
Once they have this generated image, which is usually spurious, they overlay the original over it, only using the reconstruction for the points occluded by the mask. Their findings indicate that the results in this step can be greatly enhanced by using Poisson blending [Pérez, Gangnet, and Blake (2003)].

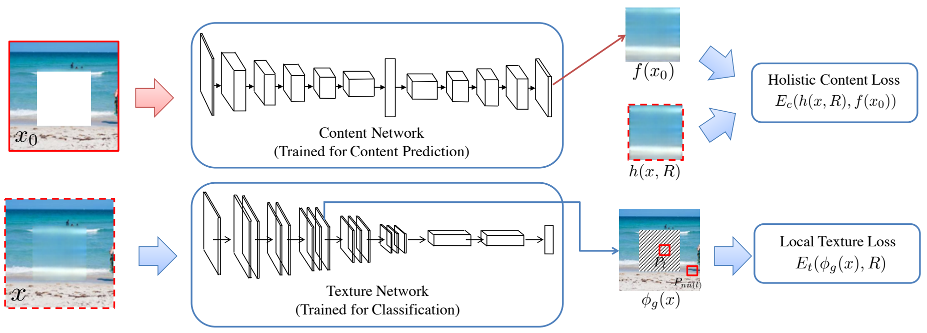


Figure 6: Architecture of High-Resolution Image Inpainting method (source:[Yang et al. (2017)]).

3.5 High-Resolution Image Inpainting using Multi-Scale Neural Patch Synthesis

One of the main objectives of this method, is to aim for semantic inpainting at high resolutions. While the works previously explored were restricted to a somewhat small resolution, 64x64 and 128x128 respectively, this works aim to have a method that scales together with the resolution.
This work also brings back ideas from previous works such as [13] in their decision to use patches. While other methods tried to focus on either, the problem of reconstructing structure heavily patterns such as textures, or reconstructing semantic parts, this method aims to have a model capable of both.

This model contains two different architectures, as it can be seen in fig. 6, delegating between two networks the tasks of knowing which content to draw and how to draw it.
The first one is the **content network** designed to create a reconstruction to use as a prior. This reconstruction should roughly resemble the desired end result, this means that this network has to be capable of learning most of the semantic information required by the problem. But, differently from the other methods, it can do so without much detail and with blurry textures.
To implement the content network, they use a variation of context encoders which replaced the ReLUs for ELUs and used a fully-connected layer instead of the channel-wise fully connected layer.
After that, a **texture network** is applied. The objective of this network is to optimize iteratively the reconstruction by focusing on generating plausible textures. To implement this texture network they used middle layers from the VGG network pre-trained for image classification. The reason for this choice is that they believe the features extracted by these middle layers are high level enough to have strong invariance to texture distortions.

3.6 Iterative Process

Once the prior image is generated by the content network, they apply the texture network iteratively to increase the results each time. In doing so they aim to optimize the following problem, having a new image $\tilde{x}_{i+1}$ that improves over the last step x_i by minimizing the following:

$$\tilde{x}_{i+1} = \arg\min_x E_c(h(x,R), h(x_i,R)) + \alpha E_t(\phi_t(x), R^\phi) + \beta \Upsilon(x)$$

where $h(x_1, R)$ is the prior generated by the content network and $\phi_t()$ is a set that contains the feature maps of the texture network t. This formula is trying to minimize three different constraints which evaluate the quality of the reconstruction. One of the interesting properties given by this approach is that it is greedy, as it always takes the best possible optimization at each step, even if that is not the optimization that will ultimately led to the best end result.
The first constraint this model is trying to minimize is the **holistic content constraint** which similar to other models is based on the ℓ_2 distance. In this case though, we are not calculating the distance to the original, but to the previous reconstruction x_i:

$$E_c(h(x,R), h(x_i,R)) = \| h(x,R) - h(x_i,R) \|_2^2$$

The second constraint is called **local texture constraint** and its objective is to correctly quantify the quality of the reconstructed textures. To do so, it splits the reconstruction into patches and calculates the activations of the texture network for each one. Then they search for patches belonging to the unmasked area which have similar activations and they use the distance as a measure. This is done with the following formula:

$$E_t(\phi_t(x), R) = \frac{1}{|R^\phi|} \sum_{i \in R^\phi} \| h(\phi_t(x), P_i) - h(\phi_t(x), P_{nn(i)}) \|_2^2$$

where $h(x, y)$ is a function to extract the subimage of x which is located on y, R^ϕ is the set of patches of the masked region, P is the set of patches of the image and $nn(x)$ is a function that finds the item on P with lower ℓ_2 distance to the patch x.
An important assumption made by this formula is that a patch with a similar texture to the one being reconstructed exists on the unmasked image. The size of the patches is static and does not change depending on the size of the mask, so this will fail when textures are smaller than the patch size.
Finally, they use a **TV loss** which stands for total variation loss.

$$\Upsilon(x) = \sum_{i,j} ((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2)$$

where $x_{i,j}$ indicates the pixel value of x at position i, j. Minimizing this constraint encourages spatial smoothness as it heavily penalizes high frequency details. Do note that the objective of this constraint is only to penalize, it should never get close to 0, as that would result in reconstructions of a single solid color.

4 Evaluation

Once we have clear knowledge about the methods for this problem, we need a way to evaluate how good the solution generated by each one is, to do so we will prepare two kinds of evaluation tasks:

4.1 Reconstruction Quality

This is one of the standard evaluation processes for inpainting techniques. It consists in analyzing the difference between the inpainted part of the blocks and the original content. We want our process to weight all images similarly, independently of the number of blocks. We also want each block to be evaluated independently of its size. To accomplish this, we will normalize the results.
To measure the quality of the solutions, we will pick a set of metrics which provide different and relevant information about the quality of the reconstruction:

1. MSE: This measure has widespread use in traditional computer vision. To calculate it, we obtain the squared difference between each corresponding pixels in both images and average them together. This results in the following formula:

   $$MSE(x,y) = \frac{1}{N}\sum_{i=1}^{N} |x_i - y_i|^2$$

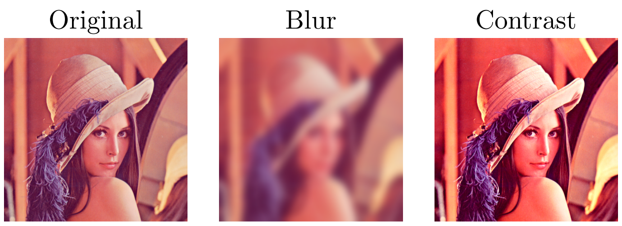
   where x_i and y_i correspond to the pixel i of each image respectively. This measure is really simple, always positive and parameter independent, which makes it a standard. It does have some problems though. It does not correlate directly to how humans perceive images as it reacts equally to all transformations. Instead, humans evaluate some transformations worse than others, an example of this can be seen in fig. 7.

2. Structural Dissimilarity Index(DSSIM)[Wang et al. (2004)]: This index is based more on how humans perceive images, that is because it uses three independent parameters: luminence, contrast and structure. To calculate it we will use a sliding window without overlapping and apply the following formula at each step:

   $$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c1)(\sigma_x^2 + \sigma_y^2 + c2}$$

   Where μ is the average of the image, σ the variance, σ_{xy} the covariance, and c1,c2 constants.

Overall, in these metrics we are trying to evaluate how close the reconstruction resembles the original and if it contains noise, blurriness or other artifacts. The trade-off for this, is that these metrics are general, contextless and non specific to human image analysis. This means they will err in various cases, one such example could be a reconstruction where the intended result is a hand, being



MSE:	531	548
DSSIM:	0.40	0.13

Figure 7: Changes in DSSIM and MSE measures when applying different transformations. Note that DSSIM does not find the contrast transformation as bad as the blur.

filled as a foot. In this case, these metrics would report a small error, but if a human was to look at the results, they would easily identify the reconstruction. To promote solutions dealing with this, we will add another metric.

4.2 Human Pose Evaluation

This metric is based on a possible real use case for this problem, and evaluates an unexplored part of the methods, the knowledge they learned about human structure and pose.
For this metric we will use performance of a current state of the art method for deep-based pose estimation over the reconstructed images. We will quantify it in terms of distances between predicted joint locations and real joint locations.

To evaluate the distance between poses we will use Weakly Normalized Joint Distance (WNJD), an adaptation of common methods such as PCP and PCK. WNJD can be calculated with the following formula:

$$\frac{\sum_{i=0}^{N} |OJoint_i - PJoint_i|}{N\,||(w,h)||}$$

Where $OJoint$ and $PJoint$ are vectors containing the original and predicted joints respectively. N is the number of labeled joints and w,h are the width and height of the image.

4.3 Stacked Hourglass Networks for Human Pose Estimation

Pose estimation has been a common challenge for researchers due to its huge utility. As so, this problem is well established inside the computer vision literature and there has been numerous challenges for it.
Some of the biggest subproblems contained into this task are: robustness to occlusion, robustness to deformation, external factors such as clothing or lighting, variance of poses and others. Early works have tackled this by using traditional

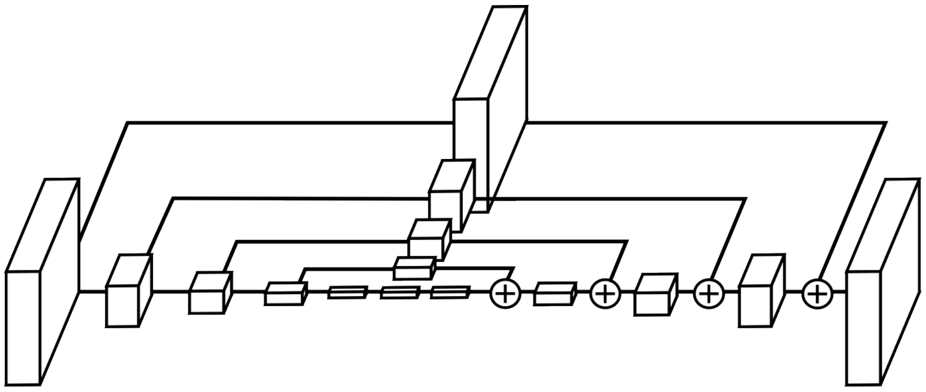


Figure 8: Architecture of a single hourglass module (source:[Newell, Yang, and Deng (2016)]).

measures such as feature engineering and structure searching [Bourdev and Malik (2009)], [Yang and Ramanan (2013)], [Ferrari, Marin-Jimenez, and Zisserman (2008)]. Since approximately 2013 this problem has increasingly been tackled with convolutional neural networks, which proved quite effectively at it [Andriluka et al. (2014)].
One of such state of the art techniques is [Newell, Yang, and Deng (2016)], which uses an architecture they call hourglasses to obtain competitive results. This is the algorithm we will use for the evaluation of pose human detection in our task. The main motivation behind the architecture they use is to capture information from all possible scales, and whereas other methods decided on processing the image at multiple scales to accomplish this, this model uses the building block they call hourglass. Basically, what the hourglasses do is branch the network after each convolution, one output of the convolution is maxpooled as it would be in a normal autoencoder, the other one is processed by a residual module to extract features from that resolution. After the bottleneck part of the architecture is reached and before every transposed convolutional (deconvolutional) layer is applied, the usual features are added together elementwise with the features obtained by the residual module at the corresponding resolution. For this model to work, the network must be symmetric as there should be a convolutional layer for every transposed convolutional one.
This architecture, pictured in fig. 8 enables them to obtain a set of heatmaps indicating possible location for each joint. Then they stack multiple of these hourglass architectures together, and when training they use the loss produced by each of the hourglasses independently. Stacking them together allows the model to iterate and improve the results while extracting higher level features.

5 Methods Analysis

In this section we will look at the different baselines and how we tuned the parameters for each one. We will also get some critical discussion about why we chose these parameters given the results obtained and the specific models used by the baselines.
The general procedure to do so will consist in using the same training, tests and validation sets prepared for the competition and choosing all the parameters by training with the training set and testing with the validation, lastly we will use the test set to check what the real results are.
For the experiments we will first explain in detail what each parameter we want to optimize does, and which range do we think we should try based on previous experiments, our intuition about the method and the original paper comments. If this is not sufficient and we think that there might be some intricacies we do not have enough information on, we will do further experiments by either increasing the range or performing different kinds of experiments.
To choose the parameters, ideally we would do an extremely deep analysis with lots of experiments and we would also check correlations between result changes with different parameters. But as these models are GANs which are notoriously hard to train, all the experiments take a long time and we will have to do as few experiments as possible. So, we will use the algorithm detailed in Alg. 1

```
Data: $Parameters \leftarrow$ List of default parameters;
$RangeParam \leftarrow$ Vector containing lists with all the values to explore for
each parameter;
$M(params) \leftarrow$ Model that given a list of parameters returns a loss value.
Result: Optimized list of parameters: $Parameters$ given the model $M$
$MinLoss \leftarrow \infty$;
for $i \leftarrow 0$to $|Parameters|$ do
    $newParams \leftarrow Parameters$;
    for $value \in RangeParam[i]$ do
        $newParams[i] \leftarrow value$;
        $DSSIM \leftarrow M(newParams)$;
        if $DSSIM < MinLoss$ then
            $MinLoss \leftarrow DSSIM$;
            $Parameters[i] \leftarrow value$;
        end
    end
end
```

Algorithm 1: Algorithm followed to optimize each model.

This algorithm makes the strong assumption that the parameters affect the results independently, this is not true for most models, but an acceptable tradeoff for the speed gained. Its performance also heavily depends on the order of the parameters given and the default parameters. If we were to start tuning parameters with a model that collapsed due to the default parameters this process would not work.

To avoid these we have chosen the default values from the original papers with small modifications, and done some experiments with each one to make sure that these parameters gave reasonable results with this dataset. For the order in which to tune the parameters, we decided on choosing it manually, according to the relevance of each parameter, we especially prioritized parameters that may cause the model to fall into different local minima.

5.1 ContextEncoders

For this model we will be tuning the following parameters: Epochs, Loss weights, Size of overlap and bottleneck size. More explanation about why we choose these variables will be specified in each section. To evaluate the performance of these models, we will use the two losses defined by this method in sec. 3.1.1 together with the loss of the discriminator. The implementation used for this model is provided by [15].

5.1.1 Epochs

First, we will start by optimizing for how many epochs we need to train the model to obtain good results. This will be a very rough optimization given that it will change depending on the other results. Even so we will optimize this parameter first due to the default value 500 being too big for the time we have, and the suspicion that it might already be overfitting. Ideally we would like to find a point with less epochs where we already have a good result, even if it is not the absolute best.
In our case, we will monitor 3 variables: Reconstruction Loss, Adversarial Loss, and Total Loss.
Training GANs is considered notoriously unstable, and the results might be hard to interpret, so we will prepare some possible outcomes and their plausible causes before executing the experiment:

- If the generator loss is continuously decreasing it is probable that the generator has found some way to reliably fool the discriminator. Ideally, the generator loss should oscillate.
- If the discriminator loss reaches and stays at 0, it probably means that the generator has collapsed. Ideally, we expect the discriminator loss to have a downwards trend and low variance.
- The reconstruction loss should have a downwards trend as the results become better, if it keeps increasing it may imply that the generator collapsed or that the adversarial loss has too much importance.

Finally, we decided on testing up to 600 epochs. We would have liked to explore how the model behaved when trained for longer, but this was a good compromise considering training time.

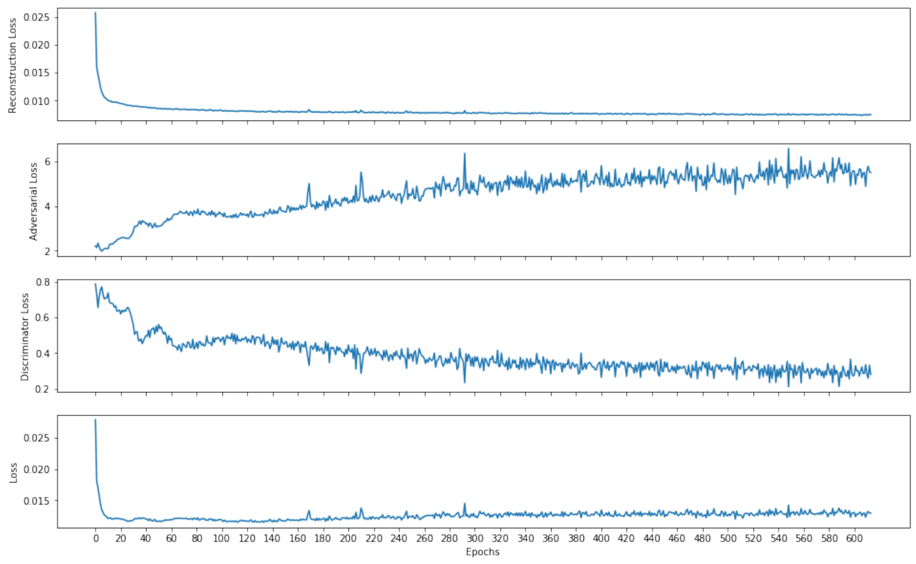


Figure 9: Losses when training the model with the default parameters.

5.1.2 Epochs Results

We can see the results of the experiment in fig. 9. We can clearly see that there was some instability until around 80 epochs, after that the trends become clearer. At first glance we cannot see any of the negative outputs we explained before, so we will check each loss individually:
The Reconstruction loss immediately falls to a value near 0 and then keeps getting lower at a very slow pace. To understand why this happen we have to remember that the Reconstruction Loss represents 0.999% of the total loss, which causes the model to focus first into minimizing this loss. This is good as it helps bootstrap the generator into the right path, then it should focus more on minimizing the adversarial loss.
The adversarial loss is slower at stabilizing, once it does it has an upwards trend until 450 iterations where its trend stop. Here it is really important to note that after 180 the results tend to have much more noise.
The immediate takeaways from these results are that the reconstruction loss might be weighted a little too high as while the adversarial loss shows that the discriminator keeps getting better at identifying fakes, the generator is not able to follow. There might be different causes for this, but considering that the reconstruction loss is so low and still going down, we suspect that it may have too much weight over the total loss. This causes the generator to prefer optimizing the reconstruction loss over the adversarial. In our assumption of how the model was working we suspected that there was overfitting due to the blurriness of the resulting images, but now we can attribute this blurriness to either the loss, or the limits of the generator. In fact, looking at this result we

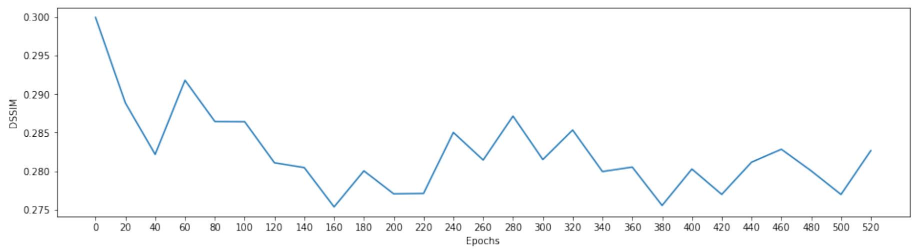


Figure 10: DSSIM on test set for models trained with different number of epochs and default parameters. X axis is not to scale.

can see that the model is still getting better at 600 epochs.
When choosing which value to use for the experiments our first intuition is to look around 150 200, as that seems to be the point where the generator starts having too much noise, which could render the experiments invalid. Even if this is not our best model, we prefer to have a model with more stable results, this is also a number of epochs which would allow us a decent tradeoff in number of experiments.
Before choosing this value though, we did some extra experiment by calculating the DSSIM of some of the models explored, the results of this experiment can be found in fig. 10. Looking at it we confirm our suspicions that while the model is getting better, there is too much instability which could be problematic. On the good side it also shows that when the model is stable (150 200) it gives good results. For the final decision we decided on **180** epochs, we believe that the DSSIM difference between 160 and 180 is too small to be reliable enough. Moreover, we think that in next experiments we might want to increase the number of parameters so we think it's better to pick a number able to accommodate these changes.

5.1.3 Adversarial loss weight

Next, we will be optimizing our choice of loss function, as all the following parameters are heavily affected by it. An explanation of how the losses on this method work can be found in sec. 3.1.1. We have to remember that these losses are trying to maximize different properties of the reconstruction, which makes choosing adequate weights for each one extremely important.
Our expectation is that increasing the weight of the L2 will push the output to resemble the original more, but due to the variance of plausible valid solutions to the problem, it will tend towards averaging the distribution, making it blurry. This is not ideal for our task as we prefer to focus on one item of the distribution even if that is not the original, thus we want to add the adversarial loss. This loss tend to get much sharper results, but in contrast to the reconstruction loss, it is more unstable, and if it is too prominent it may cause the generator to collapse.

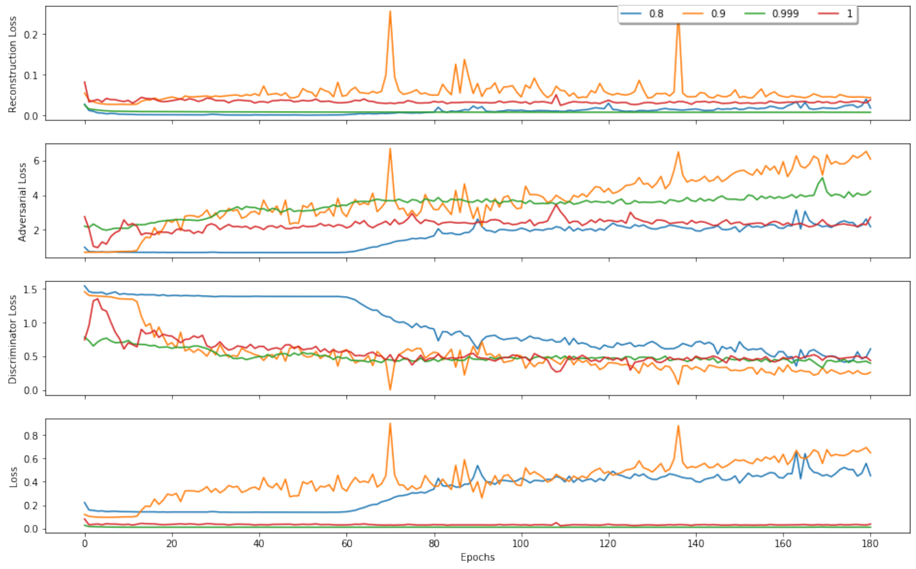


Figure 11: Losses when training models with different weights for the Reconstruction loss λ_{rec}.

On the paper they use two different values, either 0.999 or 1, both of these heavily prioritize the reconstruction loss and the latter directly disregards the adversarial loss. They recommend 0.999 but they had to use 1 due to the difficulties they faced when trying to train on the imageNet dataset with adversarial loss, where they were not able to avoid the generator collapsing.
For this experiment we will not be able to use the losses as a measure of quality as that is the part we are changing, so we will use the DSSIM on the test set as a measure.
With this in mind, we choose to test the following values [**0.8,0.9,0.999,0.1**]. We believe there is no need to test weights lower than 0.8 because of scaling. L2 is multiple orders of magnitude lower than adversarial loss, it is also absolutely necessary for the generator to converge as shown on the examples of the original paper. Our expectation is that we will find that lower L2 weights will cause the generator to collapse and cause high DSSIM. Then we will find a point where the generator is able to converge and the DSSIM is low. After that, increasing the L2 will make the DSSIM increase again. So we are mainly looking at having as high of an L2 as we can while still allowing convergence, and if this point falls near 0.8 we will do additional experiments.

5.1.4 Adversarial Loss Results

Right away we can see in the results of the experiment in fig. 11 that there are some models which had an abnormal training. The model with λ_{rec} of 0.8 had

difficulties starting up as the discriminator was not able to distinguish between fake and real results until around 60 epochs. This is in accordance to what we were expecting of models with low weights and indicates us that lower values may cause the model to completely collapse.
The model with λ_{rec} of 0.9 is clearly unstable, with lots of variance and spikes, it is not entirely clear to us why this model is less stable than the 0.8 model. The discriminator also had trouble to start converging. Even though it started earlier than the 0.8 model, we hypothesize that the discriminator may have collapsed. We consider this because the reconstruction loss is extremely high which could happen if the model was reconstructing nonsense.
The model with λ_{rec} of 1 is interesting because it did not use the adversarial loss and even then it shows to obtain a low loss compared to the others, but this should never be taken into account. That is because it was trying to optimize over the reconstruction and ignoring the adversarial loss. We reckon the discriminator latched into some feature early on which allowed it to classify somewhat well and which was not fixed due to not having feedback from the discriminator. We can see this by manually looking into the results which have a decent reconstruction for the borders but the center is just noise.
Looking at the results we have to accept that some of our initial expectations were plain wrong. We were expecting that there would be a tradeoff between the weight of the adversarial loss and the reconstruction loss, but in reality we can see that a model such as the 0.8 which almost ignores the reconstruction loss, still has lower adversarial loss than models with 0.9 and even 1.
From these results we would say that 0.999 is the best model, but it is pretty hard to extract conclusions as the total loss is calculated differently in each model and they are not comparable, so we will do an extra experiment. We can see the results of measuring the DSSIM for each model in fig. 12. The first thing we can observe is that our intuition that 0.999 was the best model was right. On the other hand, all the assumptions we had about the model when only using reconstruction loss were wrong, as we can see the model collapsed, probably due to the reasons explained previously in this section sec. 5.1.3.
From this experiment it is hard to tell exactly which is the best value, but we have strong indications that the optimal value for this exact model is probably between 0.9 and 0.999. And considering how, if we can avoid the collapse of the model, adding weight to the reconstruction loss increases the results, we can approximate that the value must be close to 0.999 so we consider it a good approximation.

5.1.5 Overlap

The parameter we will be tuning is how much overlap there should be in each block. Ideally, we would use an overlap value relative to the size of each block, but due to limitations on the structure of the network we have to use the same value for all blocks. This causes this parameter to be extremely dependant on the dataset we are working with, so we think it is important to tune it early. The original paper used a default of 4px.

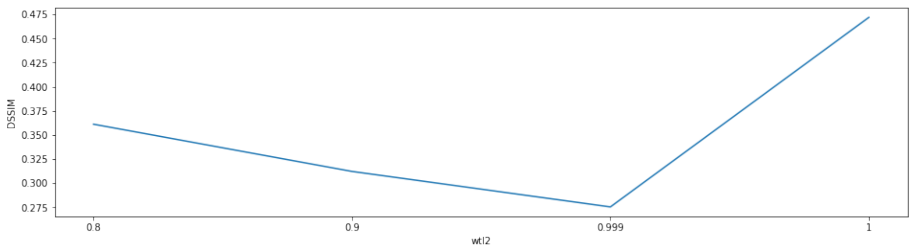


Figure 12: DSSIM for models with different λ_{rec} at epoch 180.

Our expectation is that when using small overlaps, the generator will collapse. Then at some point it will start converging, and the loss will decrease, this is the point we are looking for. Finally, at some point we will have too many pixels of overlap, the reconstruction loss will start taking precedence, and we will start generating blurry images. At this point we have too much overlap. It is also important to remember that due to how we are using the masks, we are giving extra information of the original image to models with bigger overlaps, but in the cases where we do not have enough data, we will have to add padding, and this may cause artifacts.
Finally we decided on testing the range [0,2,4,6], this range is more geared towards lower values because we consider that the reconstruction loss is already taking a bit too much weight, even with overlaps of 4. We will explore 6 just to make sure that our hypothesis is not wrong, but we expect the others to work better.

5.1.6 Overlap Results

As we can see in the results of fig. 13 the models behaved rather similarly, but we can definitely group them into models with low overlap 0 and 2, and models with high overlap 4 and 6.
Proving our initial expectations wrong the low overlap models are definitely worse, not only are their losses higher, but the reconstruction loss is stagnated. In high overlap models the reconstruction loss is slightly decreasing and given enough epochs it may decrease significantly, such as what happens in fig. 9. In the case of overlap 0 and 2 the model has possibly collapsed as the reconstruction loss is also quite high.
Our initial expectation was that we already had too much weight on the reconstruction loss, but our idea that we could decrease it and still keep a stable model was not correct, it seems that even removing a bit of it is enough to make the model collapse entirely.
For our final value we will choose 4 as it is with difference the better performing model, do note though, that even the model with overlap of 6 is not overfitting to the reconstruction loss and instead it is having problems due to the artifacts introduced by the padding.

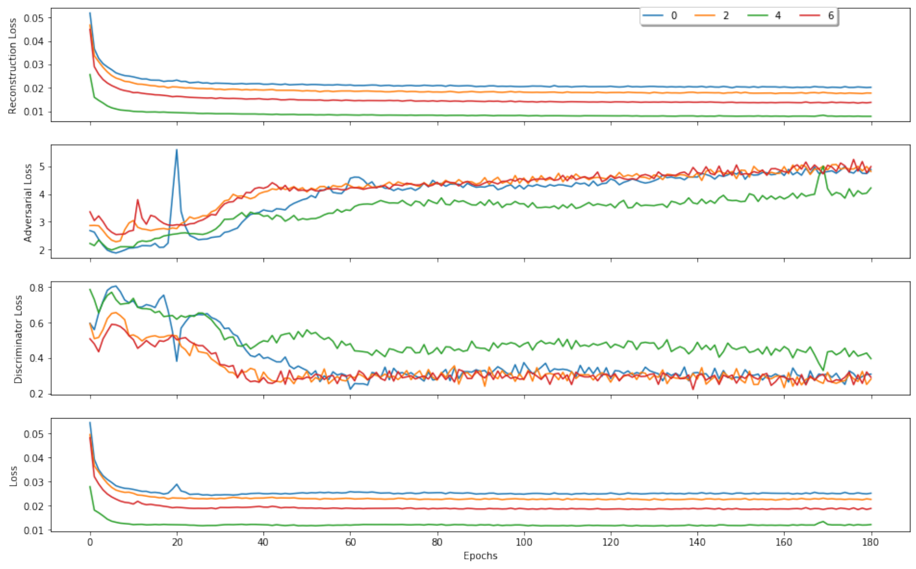


Figure 13: Loss of models trained with different overlaps.

5.1.7 Bottleneck Size

Next we fine-tune part of the architecture of the model, specifically the bottleneck. We will change the size of the context representation learned. This is an important parameter because we have to pick a vector large enough to represent all relevant semantic and low level information while keeping it small enough for the network to be trained efficiently.
The default value used in the paper is 4000, do note that this is much larger than other latent vector representation of autoencoders such as [17] (which uses 100). We believe this is due to the increase of difficulty of the task and the need to store semantic information.
For our case we think that we can have a smaller vector than the default because we consider the human knowledge needed for this task is smaller than the enormous different knowledge needed for tasks such as ImageNet. For this reason we will concentrate the range of values we will test on the smaller side, testing $[2000, 3000, 4000, 5000]$. Our expectation is that more extreme values will perform poorly, as they are either not big enough to fit the relevant information, or too big, which diminishes the implicit regulation effect. We hope that 3000 will be the better model as we believe that we need less semantic information than 4000.

5.1.8 Results Bottleneck Encoder

We can see the results of the experiment in fig. 14. From what we can see there is a clear distinction between the model with a dimension of 4000 and the

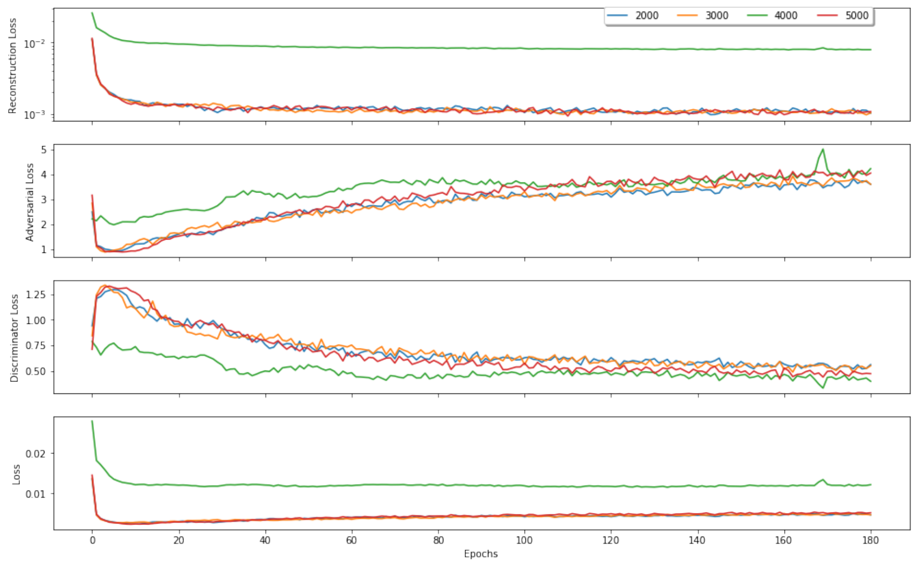


Figure 14: Loss of models trained with different overlaps. Y axis for the reconstruction loss is logarithmic.

models before and after that. This difference is specially prominent with the reconstruction loss. This might be an indicator that 4000 is the turning point between a vector too small or too large. But these losses do not tell any clear story other than probably no generator collapsed. To help interpret them we will do an extra experiment by measuring the DSSIM. This experiment can be seen in fig. 15 and it clearly indicates that 4000 was indeed the turning point. We believe that other models are outputting blurry results due to the minuscule reconstruction loss they obtained. We will choose 4000 as the final parameter.

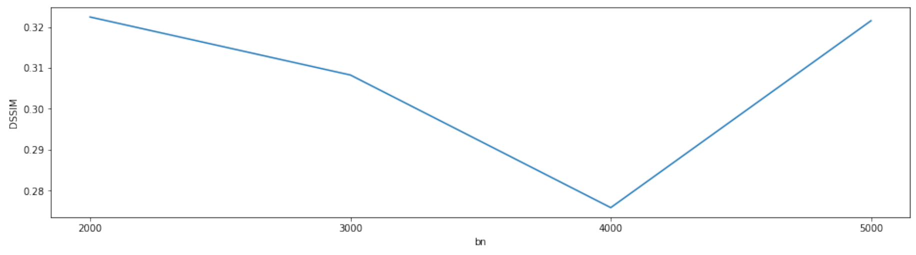


Figure 15: Loss of models trained with different overlaps.

5.2 Semantic Image Inpainting with Deep Generative Mode

For sake of keeping consistency, and due to the similarity between this method and context encoders, we decided to tune the same parameters. These parameters are Epochs, Weight of the losses and dimension of the bottleneck. In this model we do not have overlap, so we will skip it.
Sadly, due to the iterative process applied on the second part of this problem, calculating any kind of measure in the validation set is really slow. For this reason we tried to decide by only using loss and qualitative results when possible. An specific piece of information from this method, we will also use are samplings from the latent semantic space. To obtain these samplings we will create random uniform vectors and generate their reconstructions with the current model. This results in some ultimately meaningless images, but which can indicate which features the model is learning. The implementation used to train this model comes from [1].

5.2.1 Epochs

In this method we will also start by tuning the number of Epochs. The reason to do so though, is the opposite one from before. While Context Encoders had a default number of epochs too large to be able to do tests on, this model was trained with only 25 epochs. We believe this is because the original paper only had to learn how to reconstruct faces, so the problem was simpler. In our case we want the model to learn more information so it stands to reason that it should have more training time.
We are also training a GAN here so we know that the losses can be quite unintuitve. We expect not to see any of the artifacts detailed previously in sec 5.1.1, such as the generator loss continually decreasing or the discriminator loss staying at 0. We also hope too see a somewhat early point which is stable enough so that we can run our tests reliably on it.

5.2.2 Epochs Result

We can see the resulting loss while training in fig. 16, the first thing we will notice that both losses are going up. This indicates the discriminator is getting better at finding which images are generated by the generator, but it is also making more mistakes by considering images from the training set fake. To simplify it, if we consider the task of the discriminator detecting fake images we could say that its recall is improving while its precision is decreasing.
One of the reasons why this could happen is that both, the generator and the discriminator are getting better but the generator is having problems catching up to the discriminator. A less optimistic possibility is that the discriminator is being penalized too much for false negatives and is trying to compensate by assuming most of the set are reconstructions.
To help us decide which of these theories is closer to the truth we will use the sampling of the manifold at different epochs which can be seen in fig. 17. From these samplings we can see some very relevant details about how this method is

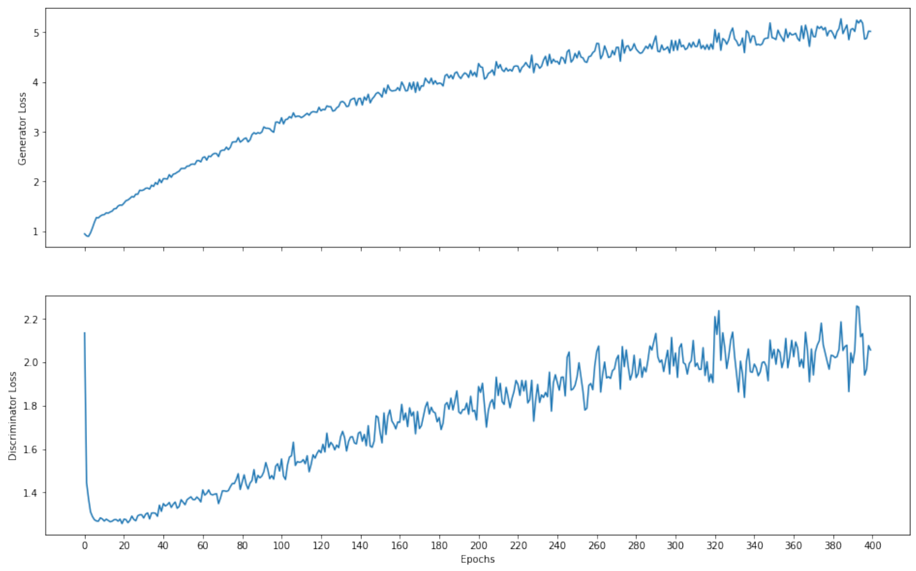


Figure 16: Loss when training semantic method with default parameters..

learning and what it is prioritizing.
We can see that it is not learning anything special about humans until the iteration 95, where legs or arms start appearing. These get further refined as the epochs increases, and at epoch 200 heads start appearing, do note that they do not have faces yet. Around epoch 400 we can see defined heads, foot and even some starting points for faces.
From this analysis we can extract that it is learning about human parts from simpler to most complicated, the most complicated being hands and faces. A similar but much faster thing happens with textures, where we can see that in epoch 25 it is starting to learn the most simple ones and by epoch 95 it has already learned a lot of them.
We can also observe how most of the positions of the vector affect the result, we can see this by observing that a lot of the samples end up being about heads at iteration 400. From this we can infer that a big space of the vector is alloted to features of heads and faces. It does so, not because there are lots of heads in the dataset, but because they are complicated. It is also an indication that it may need a bigger bottleneck.
Thanks to all these analysis, we can establish that the generator is in fact training properly. So we will decide on setting **95** as the value for this parameter. We chose exactly 95 because it fulfills some useful conditions. By then the model has learned a lot about textures and it is starting to learn about human semantics. It is also a more stable point than the subsequent epochs. Finally it is also the point where each position on the vector seem to stabilize its function.

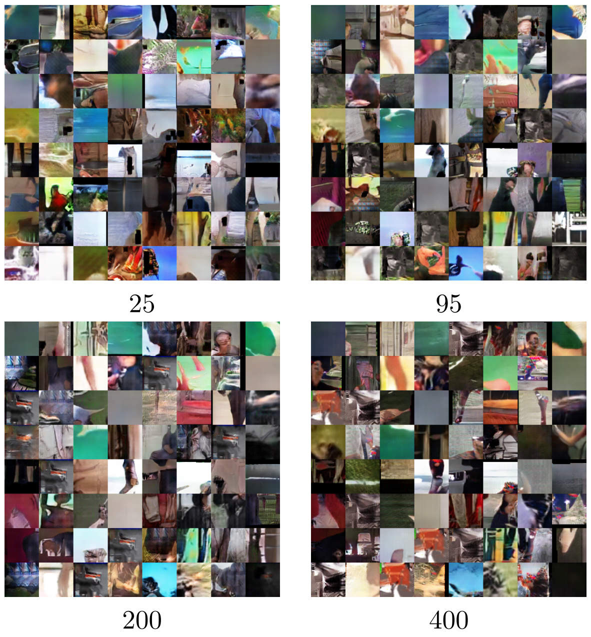


Figure 17: Sampling of the manifold space done by applying the generator to randomly generated vectors at different epochs. The vector is the same through all training.

5.2.3 Loss weights

For this model we will also tune the weights for the multiple losses, in this case the Context Loss and the Prior loss detailed in sec. 3.4.1. We will be changing the variable λ, which means that lowering the value will decrease the effect of the loss calculated by the discriminator (Prior Loss).
The default value for this parameter is 0.1, which is similar to the case in Context Encoders. Once again the simpler loss using Minkowski distances such as ℓ_1 and ℓ_2 are weighted more. This is probably for the same reasons as before, as this allows the model to be more stable while still giving good results.
The search space we choose for this variable is **[0,0.1,0.2]**. We choose 0 because we are interested in seeing how much the prior loss actually helps the model. We also want to see if by increasing the number of epochs we offsetted the instability of having a bigger adversarial loss, which could give more clear results.

5.2.4 Loss weights results

We can see the results from this experiment in fig. 18. Even though all the models have similar losses, we think that the 0.2 model is performing better. That is because we were expecting that model to be unstable, and if it is still stable with that much prior loss, it is probably giving better results. To make sure of this

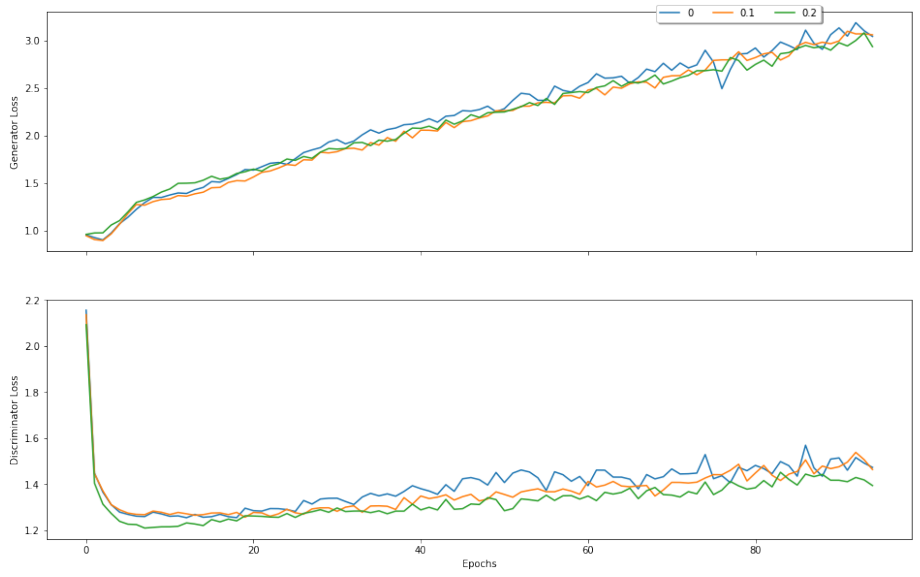


Figure 18: Loss from semantic method trained with different λ values.

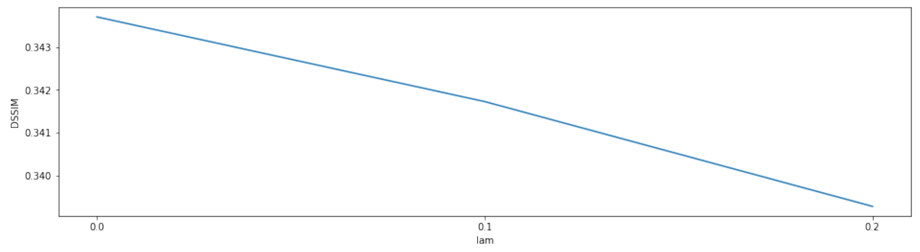


Figure 19: DSSIM for models with different λ on validation set.

we did an extra experiment and measured the DSSIM on the validation set for this models, we can see this in fig. 19. This experiment confirms our suspicions that more adversarial loss is giving us better results, ideally we would try to increase it even further, but we do not have time to do so.

5.2.5 Bottleneck

Finally we will look at the size of the bottleneck, this is extremely important for this method as the default is only 100. That may be enough for human faces, but for our problem we think we have to increase it a lot. The range we will define will be [**100,2000,4000**]. We have already seen with Context encoders that a vector of size 4000 is capable of encoding most of the semantic information needed for the reconstruction. Our expectation is that increasing the size of

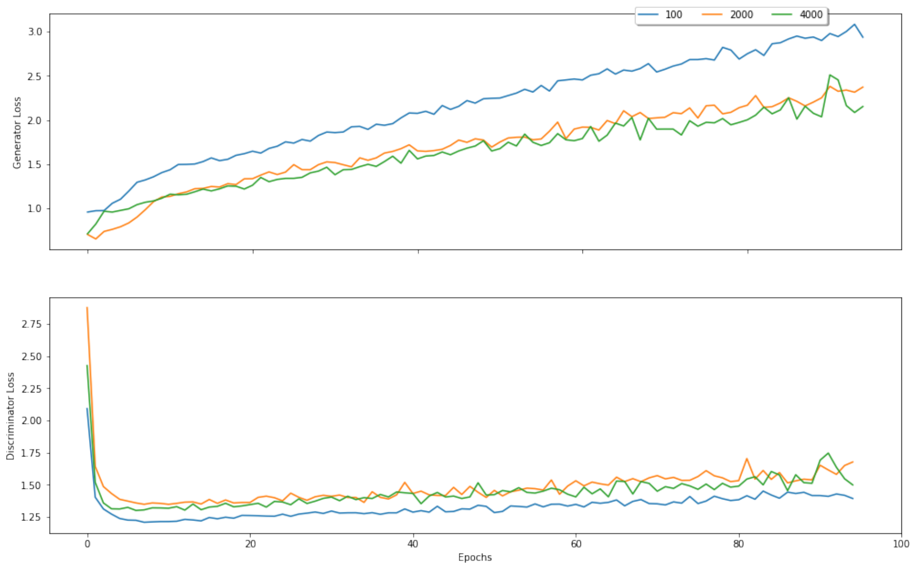


Figure 20: Loss for models with different bottleneck size.

this vector will directly increase the quality of the results. More specifically, we expect that the result is able to learn information about body parts earlier, by not having to wrestle for space on the vector.

5.2.6 Bottleneck Results

The results of the experiment can be seen in fig. 20. We can see that whereas before the generator was not able to improve as fast as the discriminator, with a bigger bottleneck the generator is able to keep up. We also can see that the improvement between 100 and 2000 is much bigger than than the one between 2000 and 4000. From this we can probably infer that there is not much difference between the two bigger vectors, and that the increase in size is mostly unused. We believe this may be due to this model learning a semantic space instead of knowing how to reconstruct from a context. This causes the model to need less semantic information as it is learning a more restricted set of possible outputs. We decided on using the parameter 2000 for because it gives almost the same results while being faster and having implicit regularization.

5.3 High-Resolution Image Inpainting using Multi-Scale Neural Patch Synthesis

This model is extremely slow inferring results due to the texture network process being long and having to be applied to each image independently. Due to this

we tried to minimize the number of testing time for this method. From the two networks of this method we only plan to train the content network, as that is the network that should contain the semantic information about humans.
Training the texture network would imply retraining the VGG network and even then, it probably would not be useful. That is because this network is only in charge of extracting features from the texture, and the domain should not have a big effect on that.
The content network in this model is the same as the Context Encoder we already optimized, so we decided on using the same parameters while training. This is based on the assumption that if the prior generated by the content network is better, the result after applying the texture network will also be better. This assumption does not always hold true, but it should hold with the vast majority of non specialized models.

5.4 Results

Once we chose the parameters for each model we did some rough experimentation with the optimizer parameters, but found out that they did not have large positive effects on the results. Once the models were trained, we generated reconstructions for all the test set images and evaluated our metrics on these reconstructions. As we can see in fig. 5.4 our metrics reveal some interesting things about how

Name	**DSSIM**	**MSE**	**WNJD**
Context-Encoders (ImageNet Model)	0.2780	0.0393	0.2673
Context-Encoders	0.2756	0.0412	**0.1489**
Semantic Image Inpainting	0.3575	0.0472	0.2427
Multi-Scale Neural Patch	**0.2530**	**0.0251**	0.1505

Table 3: Results from the test set on the methods chosen, all the models were trained with our dataset except for the ImageNet Model, which was provided by [15] and trained with ImageNet.

these methods worked. We can see that two of the methods, Context-Encoders and Multi-Scale Neural Patch worked really well for this problem while Semantic Image Inpainting had worse results.
We can also see how restricting the domain by training with our dataset containing human body parts severely increased the capability of the models to reconstruct these body parts. This can be observed through WNJD, especially in the case of the context encoders. The model trained with our dataset has similar DSSIM and MSE to the model trained with image net, but it has a much better WNJD. We believe this implies that the ImageNet model was better at reconstructing the non human parts of the image, such as background and texture. It is also important to note that the ImageNet was trained 10 times longer, which may imply that if we had more training time we could get a model capable of doing both, textures and semantic information.
We can also see how DSSIM is capable to complement MSE information. For example, while the MSE distance between Multi-Scale and Context-Encoders

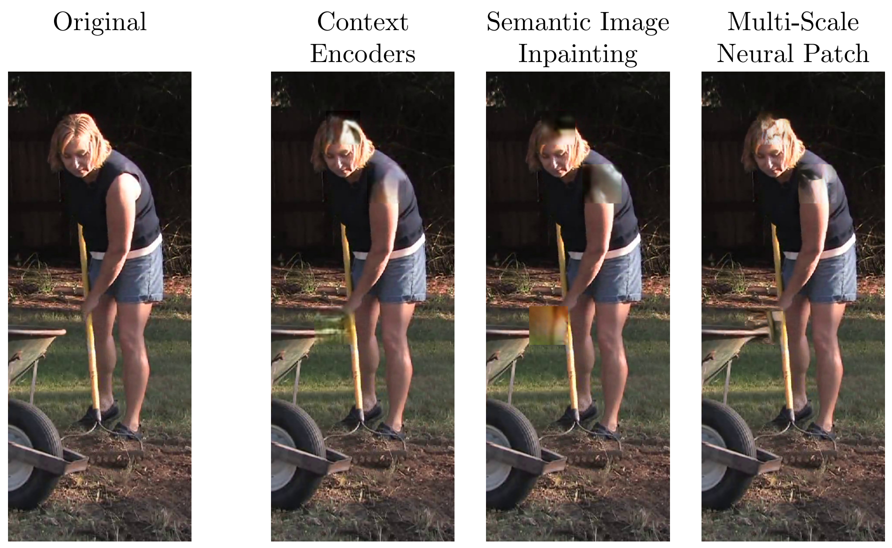


Figure 21: Example of a final result image reconstructed by all methods.

is much bigger than the MSE between Context-Encoders and Semantic Image Inpainting we can see that with DSSIM the reverse happens. In this case DSSIM is more consistent with WNJD and our own qualitative analysis fig. 21 and shows that while the solutions were different than the original, they were also valid. From these results we can see that Context-Encoders gives outstanding results considering it is the fastest method. Multi-Scale Neural Patch does not have great improvements over Context-encoders given the amount of time that it takes to apply, but we also have to consider that we are not using its full potential. In this problem we are not heavily profiting from the high resolution this model is capable of generating, which is one of the big focuses of this model.

6 Discussion

Now that we have seen an analyzed multiple methods we can clearly see that the semantic inpainting field still has a lot of room for growth. One may think that the only thing lacking for better results is more computing power to learn more semantic information. However, even in our case where we aislated the domain of application, severely reducing the complexity of the problem the results are far from perfect. We believe that there are some very specific problems which are still not strongly solved. In the methods explored we can see how each chose their own way to solve this, with better or worse results, but in no way a definitive answer. These problems are:

- **Infighting beetween objectives:** Defining what is a good reconstruction

is a really hard task. Quantify how good this reconstruction is can be even harder. In the methods analyzed we have seen two common ways to do this. Qualifying how close the reconstruction is to the original or qualifying how similar it is to other unmasked images. These two concepts are similar in nature, but they evaluate things completely different and what is good for one may not be the best for the other. We believe this problem is so important that one of the reasons why semantic inpainting is a hot topic right now is because GANs have given an effective method to calculate the later metric.
In the methods explored this problem was tackled by trying to minimize both of these at the same time, either by considering it a loss or a constraint. An example of this can be seen when changing the weight of the losses in fig. 22. This is a good way to get results currently, but it is ultimately flawed as there will be a point when decreasing one metric increases the other.
Future methods could aim to only use metrics comparing the reconstruction to other unmasked images, and completely forget about the original. This would ultimately be more aligned with what we as humans consider as a good reconstruction. If we try to do this with current methods though, we run into the next problem.

- **Instability:** This has been present mostly on the parts where we were optimizing parameters. We have seen again and again that even a small change in parameters can push these methods to collapse and not learn any meaningful information. This problem does not directly cause worse results once the method converges, but the solutions implemented to help it converge usually affect the result negatively.
  In the methods analyzed the solution to this instability was to add extra ways to start the training. This came either in simpler losses until the adversarial loss was meaningful, constrains penalizing changes or trying to help recover the model with things like ELUs. Ultimately, this is a problemQQQQ which is present on most GANs models and will probably never be completely fixed. But we wonder if we have some specific way to mitigate it for this problem. One such method could be to start the network by only using ℓ_2 loss and gradually shift the weights to the adversarial loss, or using transfer learning. Another possibility is using the advancements in the field of interpretability to detect when the model is collapsing and readjust the training automatically.

- **Onion Problem:** This problem refers to the difficulty of reconstructing parts of the image which are far away from any unmasked region. This is hard because if we decide to use the already reconstructed area in between we are extrapolating from already extrapolated data, which makes the error add up much faster. This does not only make the results worse, but also decreases the stability of the model. In the methods they tackled this problem with overlap in [Pathak et al. (2016)] and weighting the context loss in [Yeh* et al. (2017)]. In this case we believe that the second

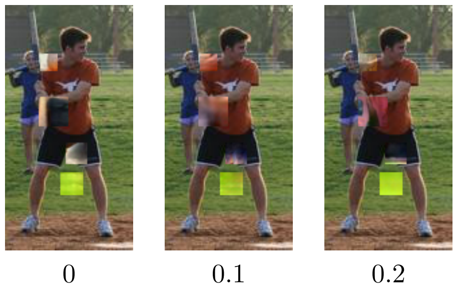


Figure 22: Example of results by the semantic inpainting methods when increasing the weight of the prior loss given by the discriminator.

method was effective at tackling this, but it could be interesting to see other approaches such as weighting applied to the adversarial loss.

We believe that solving or mitigating these problems while keeping a similar model complexity may be the way to push forward the state of the art.

7 Inpainting Competition

Part of this work will be used as a track in a competition organized by ChaLearn focused on inpainting methods. To do so the Dataset will be made publicly avaliable in `http://chalearnlap.cvc.uab.es/challenge/26/description/` together with a docker, scripts and instructions to help participant test their methods. We will also use the methods analyzed here as baselines representative of the state of the art. The competition has already been accepted as a workshop for IJCNN 2018 and it is being revised for ECCV 2018. The cometition will be ran by using the infrastructures provided by CodaLab `https://competitions.codalab.org/` Our hope is that this competition will help push the state of the art forward and find solutions to the problems discussed on sec. 6

7.1 Deploying

To simplify participating in the competition we provide a docker capable to train and evaluate all the baselines or methods prepared by the competitors.
The docker is structured on a base image containing only the necessary packets (such as tensorflow and torch) and the necessary code to evaluate the results and generate the training data. We provide a separate folder with the dataset and a folder with all the baselines, together with custom code to facilitate replicating our results, this folder should be linked to the /home/aux/ folder inside the docker.
To train the models one should use the script TrainGen.py inside the Inpainting folder to generate the data for a particular model, then moving it to the

corresponding folder. After that, the train.sh inside each baseline can be used to train it. To evaluate a method the script generate.py also contained in the inpainting folder can be used.

8 Conclusions

In this work we looked at the problem of semantic image inpainting, which consists in reconstructing occluded parts of an image while keeping semantic information as well as high level details consistent. We compiled a new Dataset themed about images of humans. To do so we used still images of humans with a high variety of poses and backgrounds. We also prepared this dataset to be used in the problem of semantic inpainting.
We established a discussion about the metrics currently used for this problem, and proposed a novel metric unique to our dataset which in conjunction with traditional metrics, allowed us to explore previously hard to measure qualities of the models such as the semantic knowledge learned. This is important because as the field is still immature, possible competitive solutions might only be actually better at improving the high level details of the reconstructions, while ultimately forgetting the semantic part of the problem.
We also analyzed the inner workings of some of the most prominent state of the art methods for this field, as well as compared their results. We proved that restricting the domain for these methods severely improves the quality of the results. We also showed how when restricting the domain, the models are capable of learning important semantic knowledge relevant to this domain, in our case human body parts.
We discussed some of the most prominent problems on the field, analyzing why they occur and how each method tackles them. This led us to find that there is no clear consensus about which is the best solution for some of these problems. Our experience allowed us to also propose some possible solutions ourselves. Finally we presented a competition in which we hope the current state of the art is pushed forward.

8.1 Further Work

While on this project we tackled the current state of the art, we think that there might be several other interesting branches to explore, here we will detail some of the ones we think are more prominent.

- **Move to a supervised problem**. We have seen that even by witholding the human pose labels while training, the model is able to perform well. It would be interesting to see models with added functionality that were able to incorporate the already known labels during training. This would greatly diminish the amount of information the model has to learn. This could even be made into an iterative process, which uses two different models improving the reconstruction and the pose estimation. First, a model similar to the one explored in this work extracts a reconstruction over which pose estimation techniques are applied. Then a supervised model improves this reconstruction via the pose estimation labels calculated, this reconstruction can be reused to extract better pose estimation labels again.

- **Model qualifying reconstructions**. One of the big problems discussed in sec. 6 is the lack of a standarized quantifiable method to value reconstructions. One possible idea to correct this is to establish a model capable of discerning reconstructions from real data and at the same time giving extra information on the reason for doing so. For example, by giving a heatmap indicating which parts of the reconstruction were weaker.

- **Human pose analysis trained with masked images**. Another interesting possible field of research is to try training the human pose estimation method on a dataset of masked data. We would like to know if the method itself would already be capable of learning how to handle this missing data. It is already prepared to be able to handle some levels of occlusion, so it would be a good exercise to know how well the occlusion handling mechanism performs. This could also be compared with our approach of applying the pose estimation method over reconstructed images.

References

[1] Brandon Amos. *Image Completion with Deep Learning in TensorFlow*. `http://bamos.github.io/2016/08/09/deep-completion`. Accessed: August 2017.

[2] Mykhaylo Andriluka et al. "2D Human Pose Estimation: New Benchmark and State of the Art Analysis". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2014.

[3] Lubomir Bourdev and Jitendra Malik. "Poselets: Body part detectors trained using 3d human pose annotations". In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE. 2009, pp. 1365–1372.

[4] James Charles et al. "Domain Adaptation for Upper Body Pose Tracking ian Signed TV Broadcasts." In: *BMVC*. 2013.

[5] Alexei A Efros and William T Freeman. "Image quilting for texture synthesis and transfer". In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM. 2001, pp. 341–346.

[6] Alexei A Efros and Thomas K Leung. "Texture synthesis by non-parametric sampling". In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Vol. 2. IEEE. 1999, pp. 1033–1038.

[7] Marcin Eichner and Vittorio Ferrari. "Human pose co-estimation and applications". In: *IEEE transactions on pattern analysis and machine intelligence* 34.11 (2012), pp. 2282–2288.

[8] Vittorio Ferrari, Manuel Marin-Jimenez, and Andrew Zisserman. "Progressive search space reduction for human pose estimation". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on.* IEEE. 2008, pp. 1–8.

[9] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems.* 2014, pp. 2672–2680.

[10] David J Heeger and James R Bergen. "Pyramid-based texture analysis/synthesis". In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques.* ACM. 1995, pp. 229–238.

[11] Sam Johnson and Mark Everingham. "Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation". In: *Proceedings of the British Machine Vision Conference.* doi:10.5244/C.24.12. 2010.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems.* 2012, pp. 1097–1105.

[13] Lin Liang et al. "Real-time texture synthesis by patch-based sampling". In: *ACM Transactions on Graphics (ToG)* 20.3 (2001), pp. 127–150.

[14] Alejandro Newell, Kaiyu Yang, and Jia Deng. "Stacked hourglass networks for human pose estimation". In: *European Conference on Computer Vision.* Springer. 2016, pp. 483–499.

[15] Deepak Pathak et al. "Context Encoders: Feature Learning by Inpainting". In: *Computer Vision and Pattern Recognition (CVPR).* 2016.

[16] Patrick Pérez, Michel Gangnet, and Andrew Blake. "Poisson image editing". In: *ACM Transactions on graphics (TOG).* Vol. 22. 3. ACM. 2003, pp. 313–318.

[17] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[18] Benjamin Sapp and Ben Taskar. "MODEC: Multimodal Decomposable Models for Human Pose Estimation". In: *In Proc. CVPR.* 2013.

[19] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning.* ACM. 2008, pp. 1096–1103.

[20] Pascal Vincent et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *Journal of Machine Learning Research* 11.Dec (2010), pp. 3371–3408.

[21] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[22] Chao Yang et al. "High-Resolution Image Inpainting Using Multi-Scale Neural Patch Synthesis". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* July 2017.

[23] Yi Yang and Deva Ramanan. “Articulated human detection with flexible mixtures of parts”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.12 (2013), pp. 2878–2890.

[24] Raymond A. Yeh* et al. “Semantic Image Inpainting with Deep Generative Models”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. * equal contribution. 2017.