Convolutional Neural Networks for Named Entity Recognition in Images of Documents

Jan van de Kerkhof

School of Science

Master's Programme in ICT Innovation

Thesis submitted for examination for the degree of Master of Science in Technology.

Stockholm 25.8.2016

Thesis supervisor:

Prof. Juha Karhunen

Thesis advisors:

M. Sc. Roelof Pieters

Prof. Tapani Raiko

Aalto University
School of Science

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Jan van de Kerkhof

Title: Convolutional Neural Networks for Named Entity Recognition in Images of Documents

Date: 25.8.2016 Language: English Number of pages: 0+45

Department of Computer Science and Engineering

Professorship: Deep Learning

Supervisor: Prof. Juha Karhunen

Advisors: M. Sc. Roelof Pieters, Prof. Tapani Raiko

This work researches named entity recognition (NER) with respect to images of documents with a domain-specific layout, by means of Convolutional Neural Networks (CNNs). Examples of such documents are receipts, invoices, forms and scientific papers, the latter of which are used in this work. An NER task is first performed statically, where a static number of entity classes is extracted per document. Networks based on the deep VGG-16 network are used for this task. Here, experimental evaluation shows that framing the task as a classification task, where the network classifies each bounding box coordinate separately, leads to the best network performance. Also, a multi-headed architecture is introduced, where the network has an independent fully-connected classification head per entity. VGG-16 achieves better performance with the multi-headed architecture than with its default, single-headed architecture. Additionally, it is shown that transfer learning does not improve performance of these networks. Analysis suggests that the networks trained for the static NER task learn to recognise document templates, rather than the entities themselves, and therefore do not generalize well to new, unseen templates.
For a dynamic NER task, where the type and number of entity classes vary per document, experimental evaluation shows that, on large entities in the document, the Faster R-CNN object detection framework achieves comparable performance to the networks trained on the static task. Analysis suggests that Faster R-CNN generalizes better to new templates than the networks trained for the static task, as Faster R-CNN is trained on local features rather than the full document template. Finally, analysis shows that Faster R-CNN performs poorly on small entities in the image and suggestions are made to improve its performance.

Keywords: Named Entity Recognition, Convolutional Neural Networks, Faster R-CNN, Images, Documents

Acknowledgements

I would like to thank Roelof Pieters for his expert supervision and help in finding a great interning position, as well as Erik Rehn and David Hallvig at Dooer AB for their supervision and for providing me with the necessary resources to complete this project successfully, and Sam Nurmi for all the good times there. I would like to thank Andrej Karpathy and Nando de Freitas for their amazing lectures, as half a year ago I knew nothing about Deep Learning. Also, I would like to thank Ross B. Girshick for making his Faster R-CNN code available and would like to thank Francois Chollet and everybody else that contributed to the Keras Deep Learning library for their awesome code. Finally, I would like to thank Marc Romeyn for convincing me to pursue Deep Learning and for the endless hours of inspiring and at times downright silly back and forth that contributed to making this work what it is.

Contents

1 Introduction

The area of named entity recognition (NER) is a region of natural language processing that involves extracting useful information from free text and dividing this information into several predefined categories, such as Persons, Organisations, Locations and Values. NER has traditionally been performed solely on a text basis, where language models are used to label entities. These are probabilistic models that are trained on hand-crafted features or rules and are currently the state of the art in the MUC-7 and CoNLL-2003 large-scale NER challenges [20, 5].

Recently, Recurrent Neural Networks (RNNs) have reported near state-of-the-art results on an NER dataset, showing that artificial neural networks can also be used for NER [17]. Moreover, Convolutional Neural Networks (CNNs), or convnets, have been shown able to understand text from nothing more than character-level encodings of sentences, obsoleting the need for handcrafted features like the ones used in traditional NER systems [32]. Convnets, in turn, have been dominating image classification and object detection for the last couple of years, breaking record after record in several large-scale image classification and object detection challenges [19, 27, 25, 8]. Convnets are big, multi-layered artificial neural networks that have been shown able to recognise and extract high level object features from images, when trained properly. Furthermore, convnets have been successful at classifying a wide range of documents [11], showing that convnets can learn so-called "landmark" features that define documents, like headings, salutations and addresses [28]. Also, it has been shown by Zhu et al. that, in the context of expense reimbursement for receipts and invoices, hand-crafted, layout based features can be leveraged efficiently to extract named regions by means of a probabilistic model [33]. While the layout features used in their work are not completely independent from the textual ones, this research shows that a lot of valuable information about the entities in a document can be extracted from the layout itself.

These findings give thought to the possibility of using convnets purely on the pixel representation of documents with a domain-specific layout to do named entity recognition. Such documents are receipts, invoices, forms and scientific documents and an NER task could consist of extracting titles, authors, addresses, amounts, etc., from these documents. Since convnets have been able to understand text solely from the character representations of sentences, they may just as easily do so based on the pixel representation of the document itself, as shown from their ability to learn high-level document features. More so, in documents with a domain-specific layout, we do not have to depend solely on textual features to do NER. Here, the network may be able to recognise entities based on a combination of textual features (characters and words) and layout features (spacing, alignment, font size, boldness). Furthermore, a small network might be able to perform just as well as the very deep networks that are used for image classification, as the network might have to learn significantly less features than networks used for 1000-way image classification. Also, the high level features that the network learns might be very different from those used for image classification, as the document domain is inherently different from that of image classification. In this work, convolutional neural networks are researched with regards to an NER task on document images. If a convnet can be successfully trained to perform this task visually, this would obsolete the need for complex NER systems like the ones developed by Zhu et al. [33], that depend on hand-crafted features for entity recognition. An automated NER system for documents with a domain-specific layout could then be developed simply by training a convnet on ample annotated data samples. This would facilitate the development of such systems and such an approach would easily scale to different types of documents, since the training procedure is similar for any type of document. Specifically, this work tries to answer the following research questions:

- How effective are convolutional neural networks in extracting a static set of names entities from images of textual documents, and what is the best network architecture for this task?
- Does fine-tuning a network initialized with learned features from an image classification task (ImageNet), as opposed to training the network from scratch, lead to better performance of the network?
- How can the dynamic object detection framework Faster R-CNN (Regions with convolutional features) be applied to named entity recognition on documents and how well does it perform compared to static CNNs?
- What is the difference between the features learned by the static networks and those learned by a Faster R-CNN network and what is the practical applicability of both methods?

The remainder of this work is as follows. First, section 2 provides a discussion of the ethics and sustainability of this works, after which section 3 provides a small overview of artificial neural networks and the history of convnets. Then, section 4 discusses the different network architectures used in the experiments and section 5 explains the dataset that the networks are trained on and the limitations of it. Section 6 discusses the experiments and the results in detail after which section 7 discusses the findings from both approaches and provides a comparison between them. Then, conclusions are drawn with regards to the research questions and the practical applicability of both approaches in section 8 and finally some suggestions for future work are given in section 9.

2 Ethics and sustainability

This work adds to the development of efficient entity recognition systems, while simultaneously adding to the research in computer vision and deep learning.

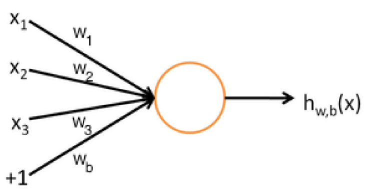
Firstly, improving the efficiency with which companies and individuals are able to perform entity recognition adds to the white collar automation that society has been seeing in the past decades. Improvements in entity recognition will improve the processing speed for documents for many different areas of industry, amongst which are accounting, law and finance. On the one side, this will clear up resources that could be invested elsewhere, making the industry more efficient. This increase in efficiency will hopefully lead to an increase in efficiency of society as a whole, as less resources will be required for many tasks. On the other side, increased automation might result in job losses for people who make a living out of annotating and processing documents. Whether this kind of automation is unethical, is something that is still a subject of discussion, and cases can be made for and against, as addressed by M. Ford in his book *Rise of the Robots* [6]. A supporting case is that it is a natural way of society to progress, improving our economic efficiency and developing a society in which life can be supported comfortably, without requiring much human effort, allowing human kind to prosper and focus on scientific progress. A case against is that on the short term, automating away white collar jobs will make the economy implode, as unemployment soars and the working class will not have enough buying power to sustain the economy. There are, however, measures to counter this implosion effect, such as a basic income. However, the discussion about this subject remains open.

Secondly, contributing to the research in computer vision and deep learning will contribute to developments in the field of Machine Learning and Artificial Intelligence (AI). An increase in AI can lead to many benefits in society, with many different applications for computer vision alone. However, there is much discussion about whether an increase in AI is good for humanity. Here, there is a fear that once strong AI is developed (an AI that is equally smart as or smarter than humans), it might obsolete human participation in society or destroy human kind overall. The technologies used in this work, however, are considered as narrow AI, and are very limited in their capabilities. They can only be used for good, or evil, once they are combined within a larger system, the impact of which is completely dependent on the system itself and the intentions of the creators, and the discussion of which is a different discussion altogether.

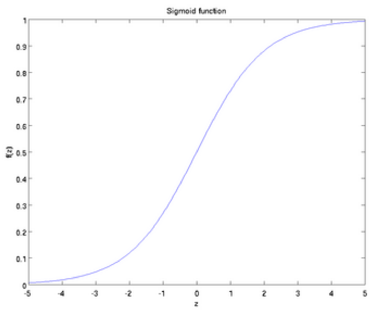
3 Background

In the last decade there have been many advances in the field of image classification and object detection in images, most of which can be attributed to research into convolutional neural networks (CNNs), or convnets. Convnets are a type of artificial neural network. Artificial neural networks, which will be referred to as neural networks from now on, are pattern recognition and classification tools that are characterised by their ability to learn from data and adapt to new training data. Neural networks are inspired by biology [15] and consist of one or multiple layers of *artificial neurons* that take real-valued numbers as input and produce real-valued numbers as output. An artificial neuron, as illustrated in figure 1a, is a unit that takes multiple inputs that are fed in through weighted connection. The neuron then outputs a value, or "fires", based on an activation function of the sum of the inputs multiplied by their respective weights.

Figure 1: Neural network fundamentals



(a) An artificial neuron [1]

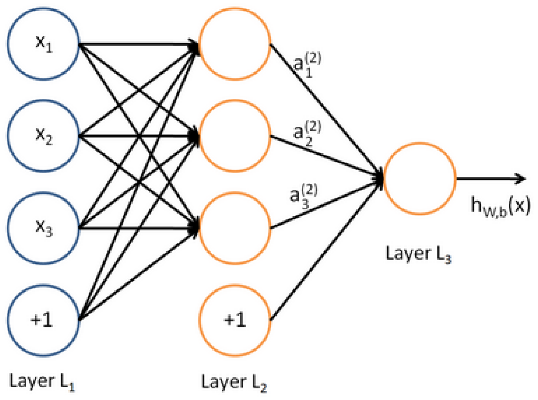


(b) Sigmoid activation function [1]

In figure 1a, $x_{1,2,3}$ are the inputs into the neuron and $w_{1,2,3}$ are the weights with which they are multiplied. The $+1$ input symbolises the *bias node*, that always inputs 1. The bias node, multiplied by its weight w_b, determines the bias b that is fed into the neuron. A typical activation function is the sigmoid activation function, where $h_{w,b}(x) = \frac{1}{1+e^{-z}}$ and $z = wx + b$, i.e. the sum of the inputs multiplied by the weights plus the bias. The bias enables the activation function to be centered around different thresholds. As illustrated in figure 1b, the sigmoid activation function mostly outputs values close to 0 or 1, where the threshold is centered around 0. The addition of the bias node makes sure the activation function is centered around the value of b, rather than 0, allowing for more flexibility.

An artificial neural network can be built by connecting one or multiple of these layers to each other and attaching an input layer and an output layer. The layers in between the input and output layers are called hidden layers. Such a network is illustrated in figure 2. The network takes the inputs from the input layer and feeds them through each layer of the network, at each step computing the activations of the neurons ($a_i^{(2)}$ here indicates the activation of neuron i at layer 2) . This forward computation of activations is also referred to as the *forward pass*. It has been shown that networks with as few as one hidden layer are able to approximate any real-valued function to any arbitrary level [16], making them great learning tools.

Figure 2: A feedforward neural network with one hidden layer [1].



The feedforward neural network is the simplest type of neural network in terms of architecture and understanding. Other commonly used types are Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs or convnets), from which the latter will be the main focus of this work.

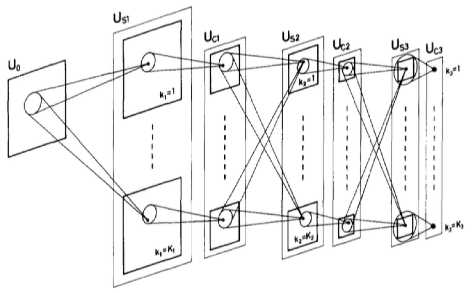
3.1 Backpropagation

The real power of neural networks comes from training by *backpropagation* [13]. Backpropagation, or backprop, is a learning technique that allows a neural network to learn from a *training set* of labeled data samples. Backprop obsoletes the need for complex handcrafted features and allows the network to learn when provided with ample labeled data. The weights of the network are then initialized semi-randomly (there are multiple methods for this) and data examples are fed in either one-by-one or in so-called *mini-batches*, where the calculations are done batch-wise to avoid expensive computation. Based on the provided labels for the data samples and the output of the network, a *loss function* computes a real-valued number indication how "wrong" the network was in its prediction. The loss function should be designed so that the more the network is wrong, the higher the loss is. Since the activation functions in the nodes are differentiable, the loss can be computed with respect to every node in the network and the weights in between the layers. By computing the loss gradient layer by layer from the output layer to the input layer, the loss is "backpropagated" throughout the network. At every backprop step, every weight is updated in the opposite direction of the loss gradient by some measure α, which is called the *learning rate*. If the learning rate is set correctly, every backwards pass should decrease the loss function by some small margin, until the loss function converges to an optimum. This process is also called *gradient descent*. There are multiple ways by which these gradient updates can be done to improve the efficiency and speed with which the network converges to an optimum. Such a method is also referred to as an *optimizer*. Most networks in this work are trained by means of the Adam optimizer [18], as this optimizer gives the quickest and best convergence. The explanation of Adam is beyond the scope of this work. The reader is referred to the literature for a detailed explanation.

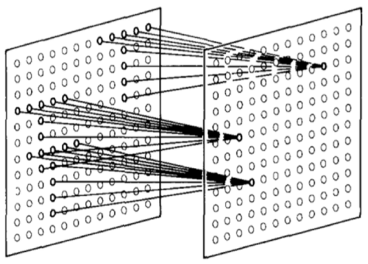
3.2 Convolutional neural networks

Convnets find their origin in the 1980's, when Fukushima et al. introduced the Neocognitron [7], a neural network model intended to do visual pattern recognition that exploits geometric similarity and that is invariant to positional shifts. The Neocognitron is inspired by the way the human visual cortex does pattern recognition. As opposed to regular feedforward neural networks [13], which consist of several layers of fully connected nodes, the Neocognitron has *convolutional layers*. Every convolutional layer of the Neocognitron consist of 2-dimensional weight vectors, or *filters*, that shift step-by-step over the input space, producing a 2-dimensional mapping of the inner product between the input space and the filters weight matrix. The network is made up out of several convolutional layers, where the first layer is connected to the pixel values of the input image and each subsequent layer is connected to the output mappings of the previous layer. This way, the filters in the first layer of the network respond to low-level features in the image, while layers deeper in the network are able to model more high-level, abstract features. The overall architecture of the Neocognitron is illustrated in figure 3a and an illustration of a weight vector is given in figure 3b.

Figure 3: The neocognitron [7]



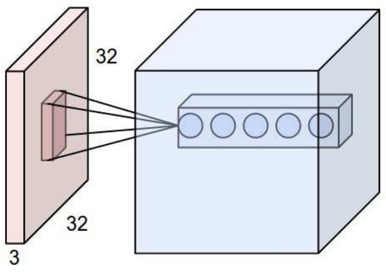
(a) Architecture of the Neocognitron



(b) Interconnection of cells between two layers of the Neocognitron

Since its introduction in 1980, several improvements have been made to the Neocognitron and multiple Neocognitron-inspired network architectures have emerged. Modern convolutional layers are no longer 2-dimensional, but 3-dimensional. Also, the image input into the network is 3-dimensional, where the depth comes from the RGB-color scale of the pixels in the image. The filters of a convolutional layer are then made up out of 3-dimensional weight matrices that span the entire depth of the output volume of the previous layer and are shifted over the volume of the remaining two dimensions, which are width and height. Each of these filters then produces a two-dimensional activation map, that is also referred to as a *feature map*, because each filter usually responds to a different feature in the image. The feature maps output by filters at the same layer have the same size and can therefore be concatenated in the depth dimension, producing a 3-dimensional output volume for each layer of the network. This output volume is then fed into the next layer of the network. An example of a convolutional layer is given in figure 4.

Figure 4: A convolutional layer [1]



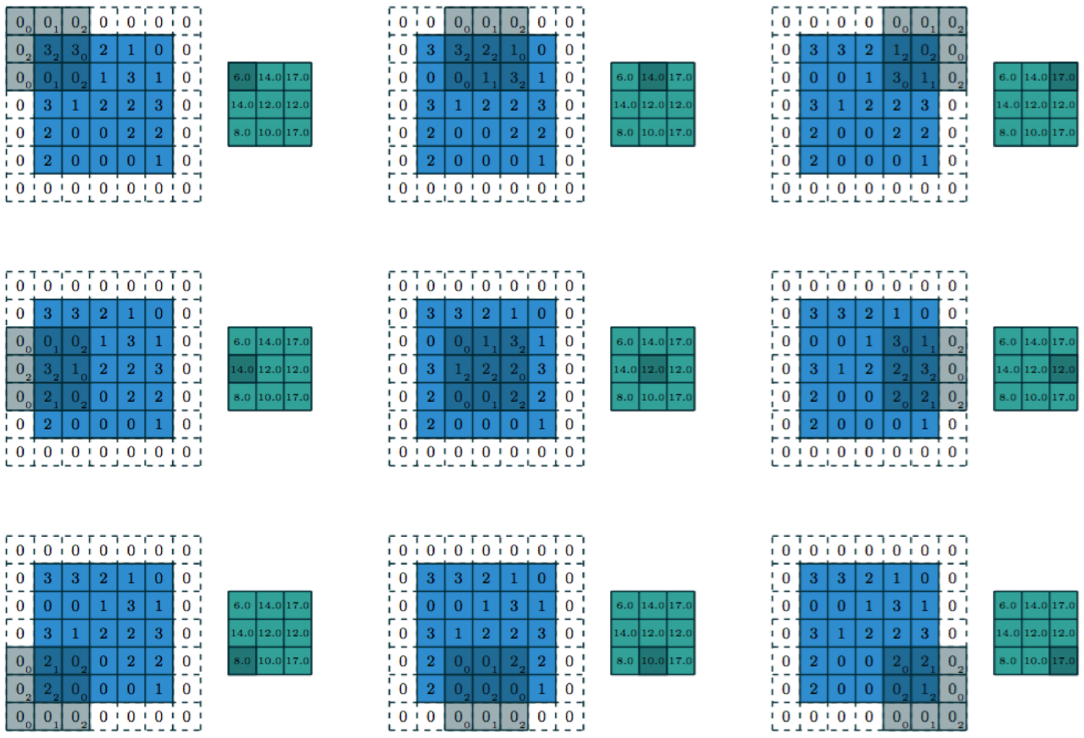
The size of the filters in a layer is called the *receptive field*. Popular choices are 1x1, 3x3, 5x5 and 7x7 pixels. One of the earliest large-scale convnets, AlexNet, had filters with an 11x11 receptive field. Large receptive fields have since then become less popular, as it has been shown that a stack of multiple smaller layers will lead to better performance. Such a stack will have the same receptive field, but will have more favorable properties [25]. For instance, such a stack contains fewer parameters while allowing for more non-linearity's in the approximation function, increasing the expressing power of the network.

The step-size with which a filter is shifted over the input volume is called the *stride*, which largely determines the size of the output volume. For example, with stride 2, the filter will compute an output every two pixels. A common thing to do is to add *zero-padding* to the input volume, to make the filter size, stride and input volume match. Figure 5 illustrates a pass of a 3x3 convolutional filter over a 5x5 input volume with a stride of 2 in both horizontal and vertical direction and a zero-padding of size 1. This pass creates an output volume of 3x3. A thing to note is that with a stride of 1 and a zero-padding of 1, a 3x3 filter will produce a feature map of the same size (in width and height dimension) as the input volume. This is used in the very deep VGG network to retain the same output volume at every step of the network, which will be explained in more detail in section 3.7.

3.3 Rectified Linear Units

A problem that arises when training deep (3 layer or more) neural networks through backpropagation is that of vanishing gradients. Vanishing gradients are a direct result of sigmoid or tanh activation functions, which have a buffering effect, meaning they get "saturated" at inputs that differ moderately from the threshold value (or bias). Once the sigmoid has been saturated, any additional increase in input will only result in a minuscule increase in output value. When the gradient is then computed with regards to the input values, the influence of each of the input values is also going to be minuscule. If these gradients are then backpropagated through several more layers, the gradient will become very close to zero, making training slow and inefficient. To address this problem, nodes in a convnet have a linear

Figure 5: A convolutional pass visualised for one filter [4]



activation with a rectifier, hence the name Rectified Linear Units, or ReLu nodes. These nodes were first introduced by Nair et al [21]. A ReLu unit has the activation function $f(z) = \max(0, z)$, where $z = wx + b$. This allows each node to express more information while also having a thresholding mechanism. The error is backpropagated through these nodes only when $z > 0$.

3.4 Max-pooling

Another layer that is frequently used in convnets is the max-pooling layer. A max-pooling layer takes the maximum activation per filter over an n-by-n region, where n is usually 2 and the stride 2-by-2. Max-pooling has been introduced to improve the robustness against spatial shifts [30], which results from taking the maximum activation per region rather then all the individual activations. The max-pooling layer can be backpropagated through by computing the loss gradient only with respect to the node that caused the maximum activation. Since the max-pooling layer causes for a loss in information spatially, the number of filters is usually increased after the max-pooling layer to make up for the compression along the width and height axis of the input volume.

3.5 Dropout

Dropout has also been introduced to fight overfitting in large networks, by preventing the co-adaptations of feature detectors to the input space [14]. Overfitting is the phenomenon of a classifier over-adapting to the training data, making it generalize poorly to new, unseen data. With dropout, some of the feature detectors (neurons) are randomly omitted during training for each forward pass. This is usually set to half of the feature detectors. This essentially transform the network into an ensemble of different networks during training and forces the network to develop a wide variety of features by which to classify the image and teaches the network to not depend upon all features for classification. During test time, dropout is disabled and the full network is used. Dropout increases the time it takes for a network to converge to its optimum and is therefore not added to all the layers in the network but only in the second-to-last layer of the network.

3.6 Classification

To do classification, convnets need to be able to go from the feature maps of the last convolutional layer to an output layer that indicates to which class the input image belongs. This is done by connecting a classification head to every node of the feature maps of the last convolutional layer (after max-pooling). The classification head traditionally consists of two ReLu layers that are fully connected to each other and one *softmax* classification layer of n nodes, where n is the number of classes in the classification task. The softmax function produces a probability distribution over all classes. Here, each class score can be interpreted as the confidence the network has that the image belongs to that class. More specifically, softmax is defined as

$$P(y_i|x;W) = \frac{e^{f_{y_i}}}{\sum_i e^{f_{y_i}}}$$

where y_i is the truth value for class i, f_{y_i} is the output value of the node that belongs to class i and $P(y_i|x;W)$ is the chance of y_i parametrised by the input image x and the network weights W. The class labels of the examples are represented by *one-hot* vectors, vectors of size n with all 0's and one 1 indicating the correct class label. The loss function that is used for softmax is the *Hinge Loss*, or cross-entropy loss. Cross-entropy is a term that comes from information theory and measures the minimum amount of information (in bits) required to go from one distribution to the other. Specifically, the Hinge Loss $H(p,q)$ between distributions p and q is defined as

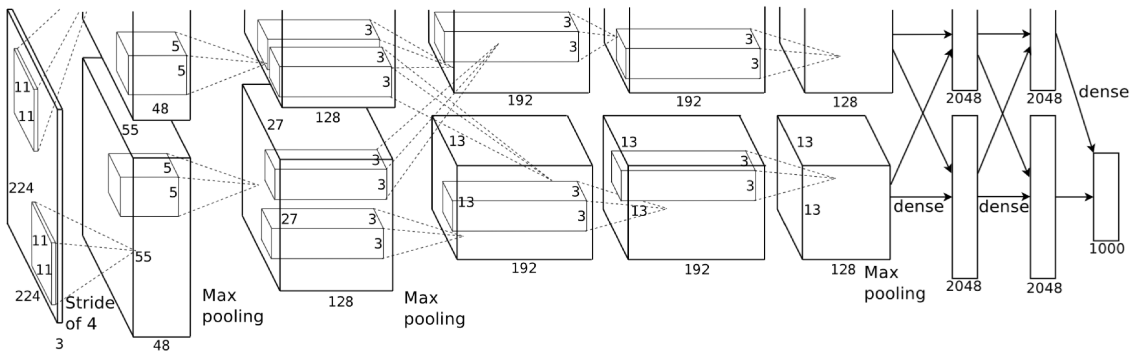
$$H(p,q) = -\sum_x p(x) \log q(x)$$

During training, this loss is used to calculate the amount of information needed to go from the softmax output to the labels one-hot "distribution". This way, the closer the network is into having 100% "belief" in the correct class label, the lower the loss will be, and vice versa. At test-time, classification is done by taking the class label that has the maximum output of the softmax layer.

3.7 Image classification

Halfway through the last decade, in 2006, a backprop trained CNN broke the record in the MNIST hand-written digit recognition challenge [22]. This was also the year that saw the first GPU-based implementation of a CNN, allowing for much faster training of these large networks [3]. Since 2012, deep convolutional neural networks such as AlexNet [19], GoogLeNet [27] , VGG [25] and more recently ResNet [12] have reported state of the art results on several image classification challenges, such as the ImageNet ILSVRC challenge, that consists of around a million images labeled in 1000 different object classes. AlexNet, illustrated in figure 6, was the first large

Figure 6: AlexNet architecture. The computation was split over 2 GPUs, which is why the image contains two convolutional stacks. [4]

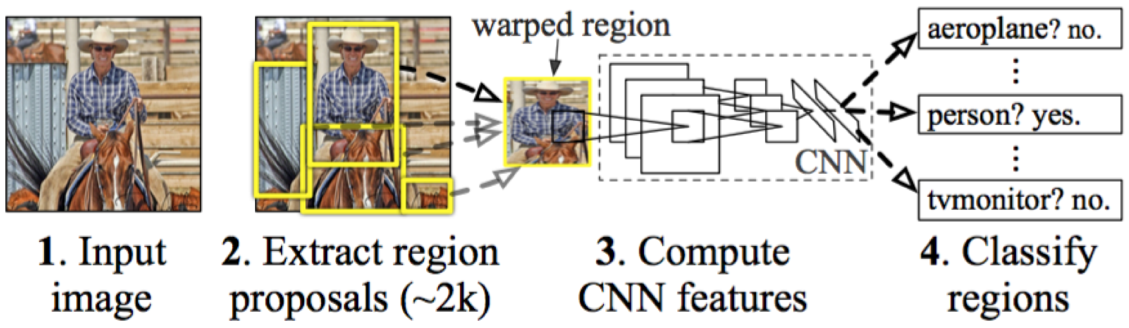


scale convolutional neural network (60 million parameters) to take first place in the ImageNET challenge. It consists of alternating convolutional and max pooling layers, followed by two fully connected layers and a softmax output layer that is used for classification. The network has 5 convolutional layers in total, out of which the first on has a very large receptive field (11x11 pixels). Large receptive fields were replaced by stacks of smaller layers, which was first introduced in the much deeper VGG-16 network, a network with 13 convolutional layers, 2 fully connected layers and 1 softmax output layer. VGG-16 has much more parameters than AlexNet, namely 138 million. GoogLeNet is a 22-layer deep network that employs a different architecture with so-called "inception modules", these are layers that simultaneously use 1x1, 3x3 and 5x5 convolutions and then concatenate the filter outputs. The motivation behind this is to be able to add more layers while keeping the computation between layers sparse, so as to not waste any computational resources. Finally, Microsoft takes it one step further with ResNet, a network that is up to 152 layers deep, that uses "shortcut connections" between layers in the network to prevent vanishing or exploding gradients, a problem that arises during backpropagation in large-scale networks. By doing so, these shortcut connections allow for much deeper models during training. While GoogLeNet and ResNet have achieved a better performance on ImageNET than VGG, VGG will be used for reference in this work because of its simplicity and wide availability in implementations.

3.8 Object detection

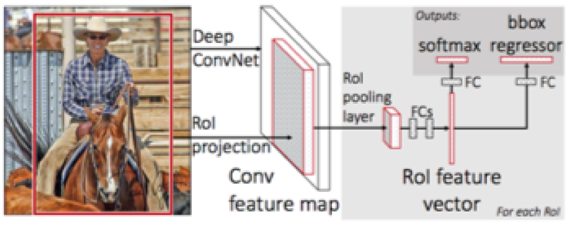
Large-scale convnets have not only been successful in image classification tasks, but also in object detection tasks. Because convnets excel at classification of an image, they have been combined with image patching and cropping schemes to perform classification on parts of images. This translates to object detection. In 2013, a system called OverFeat [24] won the ImageNet ILSVRC2013 object detection challenge, which is different from the image classification challenge, by efficiently computing thousands of different crops per image and classifying them with a single convnet (AlexNet). This record was then broken by Girschik et al. [10] when they introduced Regions with CNN features (R-CNN) and achieved around a 30% better mean Average Precision (mAP) on the PASCAL-VOC 2012 and ILSVRC2013 object detection challenges. The R-CNN pipeline is illustrated in figure 7. R-CNN uses a method called selective search to generate, for each image, around 2000 (2k) different region proposals (image crops). These region proposals are then fed through a convnet (VGG-16), which computes the convolutional features for each image. Class-specific SVM classifiers, including one for a catch-all "background" class, are then trained to classify each image crop by using the features of the last convolutional layer of the network. The increase in performance against OverFeat results from using class-specific classifiers and the more advanced object proposals of selective search.

Figure 7: R-CNN pipeline [10].



In a later version of R-CNN, called fast R-CNN [8], the image is first fed into a convnet after which for each object proposal a region of interest (RoI) is obtained from the features of the last convolutional layer of the network. By doing so, the convolutional features only have to be computed once per image, rather then once per object proposal. Once extracted, the RoIs are fed into a fully connected classification head that produces a RoI feature vector for every RoI. These feature vectors are then fed into a classification layer and a bounding box regression layer, that produces real-valued bounding box coordinates for all the classes. The softmax classification along with the regressed coordinates make up the predictions of the framework. The pipeline is illustrated in figure 8. This has been shown to be both faster and more effective than normal R-CNN, with a 9x speedup during training and 231x speedup during testing and a mAP of 66% versus 62%.

Figure 8: Fast R-CNN pipeline [10].



Finally, faster R-CNN has been proposed by Ren et al. [23], which integrates a Region Proposal Network (RPN) into the CNN. The RPN, illustrated in figure 9, replaces selective search as a way of generating region proposals. The RPN shares its weights with the CNN used for feature extraction, and slides an n x n window over the features of the last layer of the convnet, proposing k different regions at each position of the feature map. Because the RPN shares its weights with the CNN, generating region proposals becomes almost costless and thus an even bigger speedup is achieved, while retaining the same accuracy. For more details about the implementation of Fast R-CNN and Faster R-CNN, the reader is referred to the literature.

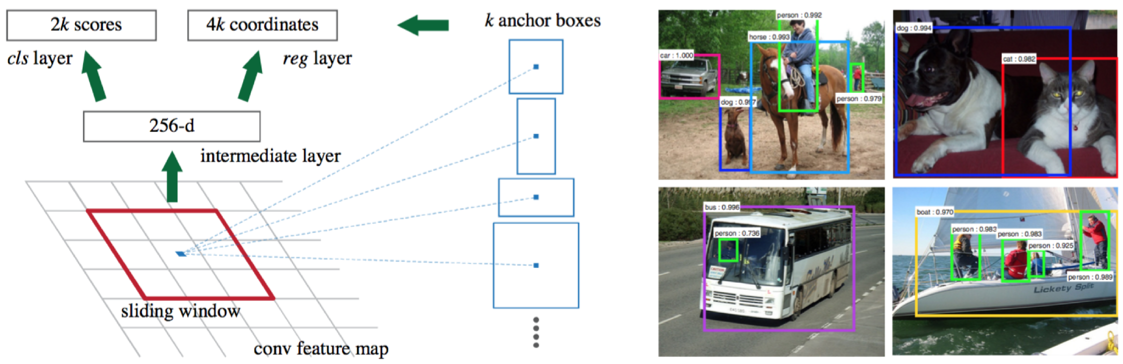


Figure 9: Image taken from Faster R-CNN paper [23]. "**Left**: Region Proposal Network. **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test."

3.9 Document classification

While NER has not been done on images of documents by means of CNNs, they have been researched in the area of document classification. Harley et al. [11] evaluate the performance of AlexNet-like CNNs with respect to a classification task on a "labelled subset of the IIT-CDIP collection of tobacco litigation documents, containing 400,000 document images across 16 categories." The authors show that CNNs perform better than methods that use handcrafted features and also show that features learned from a large-scale image classification task like ImageNET can be leveraged through *transfer learning* to improve the classification accuracy even more.

In transfer learning, instead of learning all the weights through backpropagation, a network is initialized with the weights obtained after training it on a different dataset and task, in most cases the ImageNet classification task. This training procedure is lengthy, so usually a pre-trained network that is made available by researchers is downloaded and used for this. The pre-trained network is then fine-tuned on a different dataset and task by training it with a learning rate one or two magnitudes smaller than during the initial training, in order to keep most of the learned filters intact and benefit from the learned feature extractors of the previous task. The results from Harley et al. show that through transfer learning and fine-tuning, CNNs are able to learn to recognise "landmark" features, like addresses, headers and salutations, in images of documents and use these features to perform classification.

4 Network Architectures for NER

Named entity recognition, when performed on images of documents, is essentially object detection. Object detection is done by means of outputting a bounding box around the object of interest along with a class label. This is also the way data examples are annotated in the PASCAL-VOC and ImageNET object detection challenges. In this work, NER will be done visually on documents in a similar way, where each relevant entity is annotated by means of a bounding box and a class label. Different network architectures will be compared by means of experimental evaluation, where the bounding boxes output by each network will be compared in terms of precision and recall.

Section 4.1 discusses several different convnet architectures that can be used to recognise a static number of classes in the document. The very deep VGG-16 network will be taken as the main inspiration for the network architectures and will be used as means of comparison and as network to test the effect of transfer learning. Then, section 4.2 discusses the limitations of these architectures and proposes ways to do NER by means of the Faster R-CNN framework.

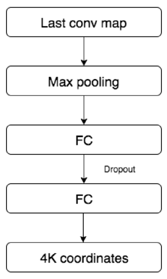
4.1 Static number of classes

One of the aspects that mostly determines the architecture of the network is whether we are looking for one specific entity in the image or whether the amount of entities that we are looking for varies. In images of receipts and invoices, for instance, there is only one date, one credit card number, one expense total, etc. When looking to recognise a *static* number of classes per image, a relatively simple network architecture can be used, where a single bounding box per entity is output. A bounding box is then defined as four real-valued coordinates specifying the upper-left x and y coordinate and the lower-right x and y coordinate of corners of the bounding box.

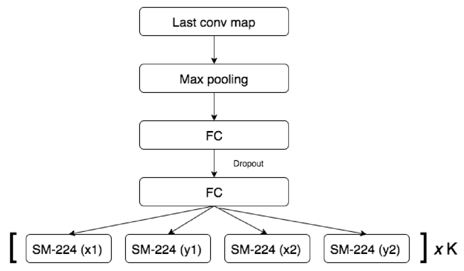
4.1.1 Transferring the features from VGG

Transfer learning has been shown to be effective in the document domain by Harley et al. [11]. The authors fine-tuned a pre-trained AlexNet to do document classification and achieved superior performance over all custom networks that they trained from scratch. The authors argue that for any classification or detection task, fine-tuning a pre-trained network should be considered as a possible solution. The same approach can be taken for VGG-16, where the pre-trained networks weights of the convolutional layers are loaded and only the classification head of the network is adjusted or replaced. Depending on the head of the network, on which will be elaborated in section 4.1.2, the network weights can be loaded until the final classification layer. Based on the results from Harley et al., such a fine-tuned network should yield the best results and will provide a good baseline to compare other solutions against.

Figure 10: Difference in fully-connected (FC) head for regression and classification for a 224 x 224 image. SM-224 stands for a softmax activation layer of 224 nodes.



(a) Regression head



(b) Softmax head

4.1.2 Regression versus softmax

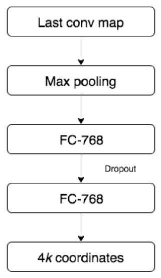
One way of outputting bounding boxes is by means of regression. instead of a softmax classification layer, the final layer of the network will be a regression layer consisting of $4k$ nodes with a linear activation, where k is the number of entities that we are looking to recognise. A typical regression loss to use for regression is the mean-squared-error loss. This is used for instance in (Faster) R-CNN to update the coordinates for each of the bounding boxes that are output. While straightforward, regression has as a downside that it is very sensitive to a small change in the input values, as the inputs into the nodes contribute linearly to the activation. This means that the network will have to find a delicate balance between feature activations in order to output exactly the right coordinate.

This sensitivity is less present in softmax classification. Because of the exponential factor in the activation function, a small change in any of the input values will not lead to a big change in activation. This allows the network to put more confidence in a certain class without influencing the outcome completely. When given a constant image size of n-by-m, the network can be "tricked" into performing classification instead of regression. This can be done by replacing each regression coordinate by a softmax activation layer of size n or m, for an x or a y coordinate, respectively. The network can then output the coordinates of the bounding box by "classifying" that the coordinate belongs to a certain pixel. This architecture will result in four softmax layers per bounding box, two of n and two of m nodes. A downside to the softmax activation function and cross-entropy loss is that there is no extra penalty when the network is off by a larger margin, as there is with the mean-squared-error loss, where the difference between the output value and the true value is squared. The difference in architectures is shown in figure 11.

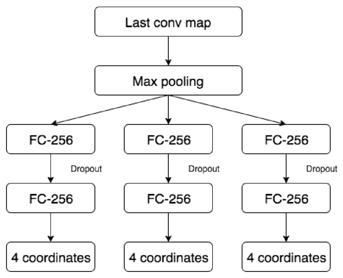
4.1.3 Fully connected heads

In terms of network heads, two different architectures are considered. The first architecture is the traditional architecture, where a single fully-connected head is appended to the last convolutional layer, after max-pooling. When the network is trained with one head, all classes share the same parameters and the fully-connected

Figure 11: **Left:** a single-headed regression architecture that has 768 nodes per FC-layer. **Right:** a multi-headed regression architecture that has 256 nodes per FC-layer. The networks have around the same number of parameters.



(a) Single-headed



(b) Multi-headed

head is trained simultaneously for all classes. When outputting bounding boxes, this architecture might make it hard for the network to create a clear distinction between the features used for defining each bounding box separately.

An alternative architecture would be to append a smaller fully-connected head per bounding box with its own, separate set of parameters. Then, only the convolutional layers are shared between the classes. Because each fully-connected head is then trained class-specifically, it seems reasonable that such a head needs far less parameters than an all-class fully-connected head.

Most of the parameters in the network are in the connection between the final convolutional layer and the first fully-connected layer(s), as all of the filters have to be connected to all of the nodes in the fully-connected head(s). For instance, for VGG, 102 million of its 138 million parameters are contained here [1]. Whether a multi-headed architecture leads to more parameters is thus dependent on the amount of classes and the number of parameters per fully-connected layer.

4.1.4 Layers, filters and image resolution

There are two motivations as for why an alternate layer architecture might perform equally well or better compared to VGG-16. The first motivation is that the input domain is limited to images of black and white documents. This domain is less varied than the ILSVRC domain that consists of 1M images in 1000 classes, in terms of the low-level features that the network needs to be able to extract. Also, the high-level features might not need to be as complex as those in VGG, where some nodes have been shows to recognise complex concepts such as reflecting surfaces [31]. Because of this, a network with significantly fewer filters and layers might be able to perform as well or better than a very deep network with many filters. The second motivation behind having fewer layers in the network is that for an NER task on images, fine-grained spatial information about the position of the entities in the document is required. Intuitively, this information is contained in the width and height dimensions of the final feature map of the last convolutional layer. With VGG-16, the image is first warped to 224x224 pixels before it is fed through the network. As a result, the final feature map is only 7-by-7 in the width and height dimension, with a depth of 512 filters. This means that the spatial information is reduced by a factor 32, which might result in valuable positional information being lost. One way of increasing the final feature map spatially is by reducing the amount of max pooling layers in the network. This means that either the convolutional layers will have to be stacked in bigger stacks (3, 4 or 5 layers) in between max pooling layers or that the network will have to contain less convolutional layers in total. A second way to retain more spatial information is by increasing the resolution of the input image. Increasing the size of the final conv map will greatly increase the number of parameters in the network. This leads to a longer training time and a slower convergence.

4.2 Dynamic classes

The architectures discussed so far have a strong limitation, namely that they only work for a predetermined (static) number of class occurrences. When looking for an entity that can occur multiple times or not at all, like for instance a line item in a receipt, this becomes problematic. A suboptimal way to solve this problem is by outputting many bounding boxes per class to "buffer" the amount of possible class occurrences, where each bounding box has an objectness score to indicate whether the class is present or not. This would, however, make the network unnecessarily large and would not scale well to many different classes. A better solution would be to use an object detection network like Faster R-CNN (section 3.8), which dynamically detects and classifies regions of interest in the input image. This way, the network architecture remains the same regardless of the amount of classes that are being classified. Also, the amount of bounding boxes output by the network becomes dynamic. Faster R-CNN's default convnet is VGG-16, pre-trained on ImageNet. Because the network is pre-trained on ImageNet, it will have to be fine-tuned towards the document domain before the high-level features become very meaningful. This is also done by Harley et al., who fine-tuned a pre-trained AlexNet to do document classification [11]. While the training procedure for faster R-CNN involves fine-tuning the convnet together with a Region Proposal Network (RPN), the training procedure is different from regular fine-tuning, as the RPN is trained jointly. This may result in VGG-16 not being able to adapt properly to the new problem domain, as the transition to the new task becomes too large. instead, replacing VGG-16 with the fine-tuned AlexNet from Harley et al. might lead to better performance, as the network will already have been to recognise document features.

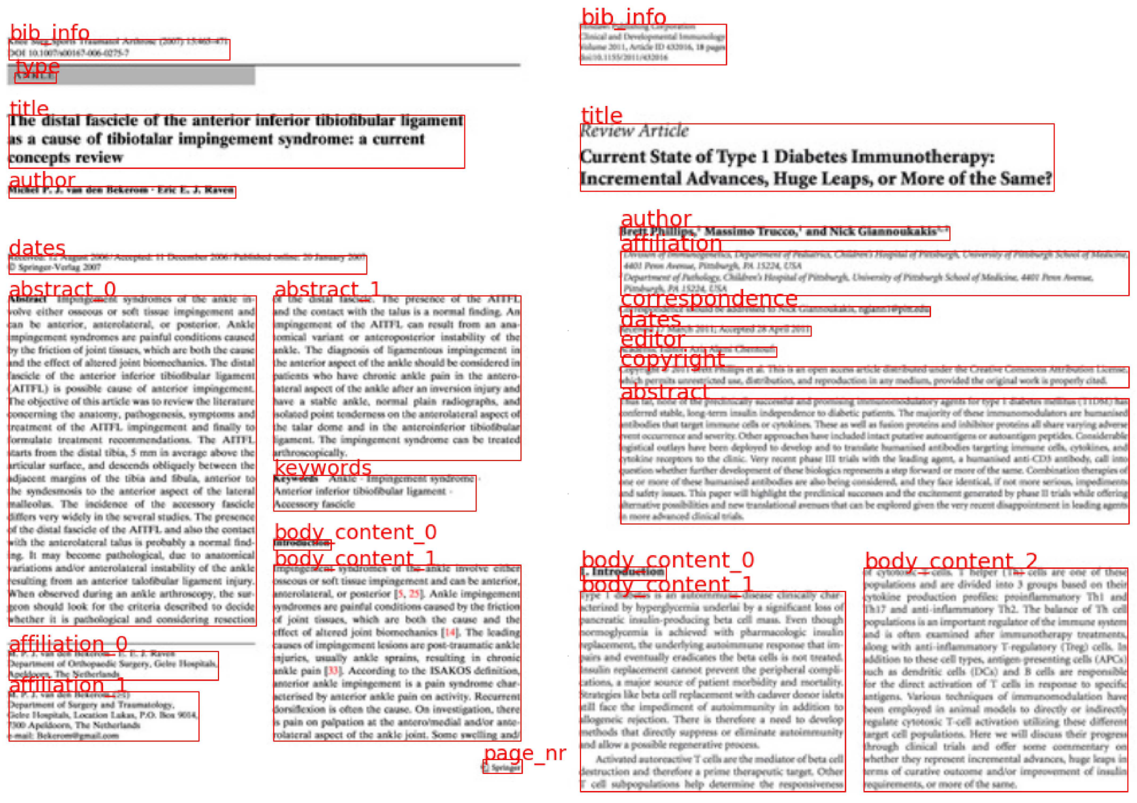
5 Dataset

To train the networks for entity recognition, an annotated set of documents with consistent entities and a domain-specific layout is required. Optimally, this would be a dataset of receipts or invoices, as in the work of Zhu et al. [33]. Here, a lot of the entities are approximately the same size and the network will have to learn to distinguish them based on the layout of the documents and the content of the entity. To the author's knowledge, this dataset, or any like it, is not publicly available. Therefore, the Ground Truth for Open Access Publications (GROTOAP2) dataset of ground truth annotated scientific documents [29] has been used for this work. From the authors' personal inspection, it seems that scientific documents are a good candidate when it comes to a correlated layout and a consistent set of entities, as each publisher has its own template and resulting layout, but almost every document contains the same set of entities. GROTOAP2 contains 13,210 life sciences publications from 208 different publishers. The ground truth of each document has been created by Tkaczyk et al. by performing Optical Character Recognition (OCR) on each document and classifying the "zones" in the document into 22 different zone classes, by matching the metadata of each document to the content obtained by the OCR. Each of these document zones indicates a textual entity, like title or author, that is present in the document by annotating the zone with bounding boxes. The annotations are hierarchical and go as detailed as one bounding box per character, but for the purpose of this work only zone-level bounding boxes are used. The documents have been annotated algorithmically by comparing each of the zones to the available metadata by means of classifiers and heuristics. Human experts have evaluated the results by taking a random sample of 50 documents and thus determined the accuracy and recall of the overall zone classification, which are around 95% and 91% respectively. The annotation is not flawless but still serves the exploratory purpose of this work. Most of the variation in zones and layout is contained on the first page of the publication, so only the first page of each document has been used for training. Figure 12 shows two examples of annotated front pages of documents contained in the GROTOAP2 dataset.

5.1 Data augmentation

Since the data set is small (13,210 documents), the networks are prone to overfitting when the data is fed in without any alterations. Early experiments, that have been omitted here, have shown that overfitting occurs when the data is fed in without augmentation. Therefore, in all the experiments in this section, each minibatch has been augmented by randomly shifting the documents horizontally and vertically across the image plane. Pixels that are shifted out one side of the image are "rolled" on to the other side of the image. The amount of pixels by which the documents are shifted is limited by how far the document can be shifted whilst keeping all relevant entities and bounding boxes intact (not rolled to the other side). Specifically, early experiments have shown that the networks converge best when the maximal amount of random shift in both the horizontal and vertical directions is set to 15% of the

Figure 12: 380x500 pixel images of the first page of publications with annotations taken from GROTOAP2 [10].



maximal distance in either direction. This is done for all the static experiments.

5.2 Normalization

After the augmentation step, it is custom to perform a normalization step before feeding the images into the network. For VGG-16 and AlexNet, this step consists of subtracting the mean RGB pixel activation values for each of the respective pixels in the image. In experiments that involve transfer learning from VGG, this normalization step is replicated for compatibility. For all other experiments, initial experiments have shown that the trained networks obtain the best results when the documents are first converted to greyscale and then each image is scaled so that all the pixels in the image have zero mean and unit variance per image. This process is also called *whitening*.

5.3 Filtering the dataset for static classes

To train the networks to extract a single entity per class, all of the documents in the training and test set need to have that class available as a singleton. Entities in the document are often annotated in a fragmented manner, as can be seen from the annotation of the *body_content* zone in figure 12. So, for experiments with static classes, zones of the same class are merged together if their merged bounding box does not overlap with any of the other classes. Still, for many classes the number of times they occur in the front page of the document varies greatly, even after merging zones. The most consistent singleton classes are title, author and abstract. Thus, in the context of static classes, the networks have been trained solely on these three classes. It occurs sometimes that a document has none or multiple title or author annotations. This is due to the document template itself or due to faults in the annotation, in which case the document will not be used. In the case of abstract, it happens often that there are two annotations, as the abstract is often split across two columns. An example of this can be seen from the left document in figure 12. In this case, the network is trained to only classify the leftmost column of the abstract, to keep the number of workable samples in the dataset to a maximum. Overall, this filtering reduces process reduced the amount of workable samples for static NER in the dataset to 10,461 as opposed to 13,179 documents for the dynamic experiments.

6 Experiments and results

This section describes the experiments that have been performed and the corresponding results. The experiments shall be discussed in the order in which they were performed, in an alternating chain of experiments and results. Firstly, the way the performance of the networks is evaluated is discussed in section 6.1. Secondly, the set of experiments that concern static classes is discussed in section 6.2. Lastly, the set of experiments regarding dynamic classes and Faster R-CNN is discussed in section 6.3.

6.1 Metrics for evaluation

The crossentropy and mean squared error loss used for training the networks serves as a means of determining how "close" the network is to outputting perfect bounding boxes, but says nothing about how well the network actually extracts each type of entity. Therefore, it makes sense to measure network performance in terms of how many words of the entity the network is able to classify correctly. A predicted entity is then defined as the words that are contained within, or intersect with, the predicted bounding box. Since the ground truth for all the words in the document is known, the performance of the network can be measured in terms of precision and recall for every class for every document. So, to measure network performance, the average precision and recall are taken document-wise over all the entities in the test set.

6.2 Static classes

The experiments in this section (and the data augmentation steps) have been implemented by the author in Python with help of the `numpy`, `scipy` and `scikit-learn` libraries. The implementation and training of the convnets has been done by means of the Keras deep learning library[1].

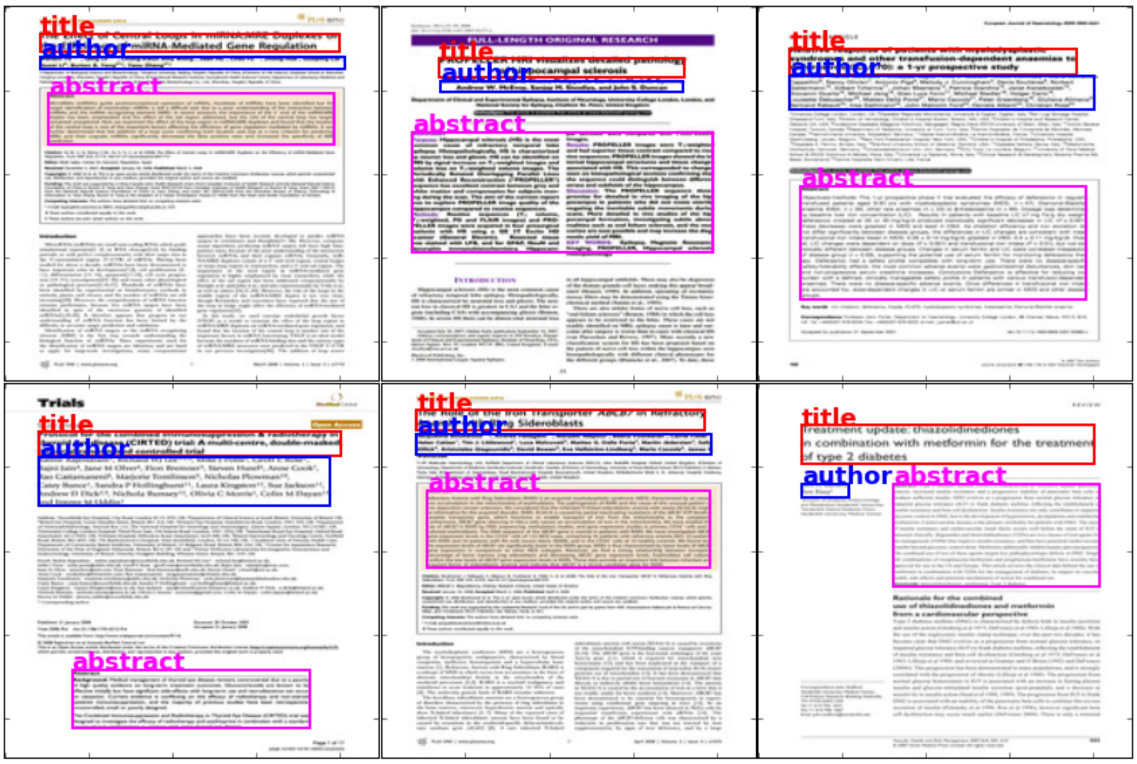
To test the performance of convnets with regards to NER, several different parameters have been firstly been evaluated in a random search and then in a couple of fine-tuned grid searches. The main architecture for the networks has been based on the VGG-16 network. All of the networks that have been tested consist of alternating stacks of 3x3 convolutional layers and max-pooling layers, followed by one or more fully-connected heads. For clarity, the heads will from now on be denoted as {single | multi}-n-{softmax | regression}. Here, single or multi indicates whether the network has a single head or multiple heads. n indicates the number of nodes per fully-connected layer per head and softmax and regression indicate the output format and training loss.

The initial set of experiments have been performed to test the influence of many different hyperparameters on the performance of the network. Since these hyperparameters are heavily correlated, an initial random grid search has been performed to determine reasonable values for some of the hyperparameters so that

[1]http://keras.io/

they could be frozen for the rest of the experiments. For brevity, the details of these experiments are omitted and only their results are briefly reported on. All networks have have been trained to convergence by following a scheme where the learning rate is halved if the validation loss has not improved for 5 epochs. The learning rate is then frozen for a minimum of 3 epochs. Training stops when the validation loss has not improved for 15 epochs or the learning rate is decreased by a factor of 2000. To keep computational time down and to keep the experiments compatible with VGG, the networks were trained on images that were warped to 224x224 pixels. Each experiment has trained the network until convergence, which takes approximately 12 hours on a single Nvidia Titan X GPU. Run-time fluctuated by up to 3 hours, depending on the network, since the exact run-time of an experiment mostly depends on the number of parameters in the network. Due to computational limitations and time restraints, each experiment has been performed once, on a single train-validation-test split of 80%, 10% and 10%, respectively. The split and random seeds have been kept consistent throughout the experiments. As mentioned in section 5.3, the networks were trained to extract one entity each of title, author and abstract.

Figure 13: Example predictions from an experiment with static classes.



From the initial random search, the optimal way of performing data augmentation and normalization, as described in sections 5.1 and 5.2, have been determined. Training the network with the Adam optimizer [18] and a minibatch size of 8 was found to lead to good results. Also, a good amount of filters for the custom networks was found, which was set at 50 filters for the initial layers of the network, doubling in size after every consecutive max-pooling layer and with a maximum of 400 filters for the final layers. This is very close to the amount of filters that VGG has and could possibly be set lower to reduce network size, but 50 has been chosen as a safety margin to make sure that a lack of filters does not affect performance. From the initial random search it was found that a multi-n-softmax head gave the best results and setting n to 256 worked better than 128 or 512.

6.2.1 Number and arrangement of layers

The first hyperparameters that were extensively tested in a grid search are the number of layers and the arrangement of layers. This was done by freezing the head of the network to a multi-256-softmax and keeping the rest of the parameters fixed as mentioned earlier, so that a good base network could be obtained on which the different heads can be compared. The number of layers were tested from a minimum of 5 layers to a maximum of 11 layers, with the first stack of layers containing 2 convolutional layers, followed by 1-5 stacks containing either 2 or 3 convolutional layers. Each stack of network layers has a 2x2 max-pooling layer in between. For reference, VGG-16 has 13 convolutional layers with a 2-2-3-3-3 layer stack, with a max-pooling layer in between each stack. The results are shown in table 1.

Table 1: Results showing the average precision (AP), average recall (AR) per class and mean average precision (mAP), mean average recall (mAR) and the F1 measure per arrangement of network layers. The units are shown in percentages.

Layer stack	AP Author	AR Author	AP Abstract	AR Abstract	AP Title	AR Title	**mAP**	**mAR**	**F1**
2-3	96.20	94.08	95.93	92.91	98.65	98.76	97.44	95.83	96.63
2-2-2	**97.18**	94.85	96.86	94.07	98.29	98.57	96.93	95.25	96.08
2-3-3	96.49	95.15	96.84	94.62	98.57	98.90	97.30	96.22	96.75
2-2-2-2	96.07	94.96	**97.17**	95.12	98.78	98.88	97.34	96.32	96.83
2-3-3-3	96.85	**95.56**	**97.17**	**95.27**	**98.97**	**99.09**	97.66	**96.64**	**97.15**
2-2-2-2-2	**97.18**	95.41	97.16	94.91	98.72	98.81	**97.69**	96.38	97.03

The results show that for almost all of the classes, the 2-3-3-3 stack has the best precision and recall, and has the best F1 measure overall. This is also the setup that has the most layers. These results invalidate the hypothesis that having more spatial information in the last feature map increases performance, as can be seen from the comparison of the 2-3-3 architecture versus the 2-2-2-2 architecture. Both networks have the same number of layers, where the latter has an additional max-pooling layer, decreasing the spatial output by a factor 2. This is also the network that has better performance.

6.2.2 Network heads

Since the 2-3-3-3 layer layout worked best for the custom networks, this arrangement of layers was frozen when performing a grid search for the performance of network heads. Softmax and regression heads were experimented with in both a single as multi-headed arrangement. For the single-headed networks, the amount of nodes per layer was set to 768, as this is 3 times the amount of nodes that was found optimal for the multi-headed version in the initial random search, which was 256 nodes per layer. This way, networks with similar size but different arrangements were compared against each other.

Table 2: Results showing the average precision (AP), average recall (AR) per class and mean average precision (mAP), mean average recall (mAR) and the F1 measure per network head on a network with 2-3-3-3 layer arrangement.

Type and number of heads	AP Author	AR Author	AP Abstract	AR Abstract	AP Title	AR Title	**mAP**	**mAR**	**F1**
single-768-regression	88.44	88.43	94.94	89.87	97.42	97.70	93.60	92.00	92.78
single-768-softmax	95.21	93.44	96.14	94.71	98.43	98.54	96.59	95.57	96.08
multi-256-regression	2.66	15.15	17.6	2.45	45.09	45.42	21.80	21.01	18.03
multi-256-softmax	**96.85**	**95.56**	**97.17**	**95.27**	**98.97**	**99.09**	**97.66**	**96.64**	**97.15**

These results show that the multi-256-softmax outperforms all regression setups by a large margin and also leads to better performance than a single headed softmax network of comparable size. This shows that the network benefits from having a seperate set of parameters per bounding box. The poor performance of regression is most likely to blame on the sensitivity in the activation function, where the output needs to be balanced between nodes, especially during training time when half of the activations is eliminated through dropout. Potentially, this is the reason why the multi-headed approach works so poorly, because the lack of nodes in the heads makes the activation much more volatile, as each node has a bigger relative contribution to the activation.

6.2.3 Transfer learning

To test the efficiency of transfer learning against the custom networks that were trained from scratch, VGG-16 was loaded with pre-trained weights on the ImageNET ILSVRC task and fine-tuned on the new NER task. Transfer learning was tested in multiple ways. Firstly, the weights of the network were loaded up to and including the final two fully-connected layers of 4096 nodes of the original classification head, after which the 1000-way softmax layer was replaced by multiple softmax layers, making the network have a single head. Since most of the parameters of VGG are contained from the last convolutional layer to the last fully-connected layer, it makes sense to transfer these weights along with the feature extractors, as they might contain valuable information about how to do classification properly. Secondly, the weights of VGG were loaded until the last convolutional layer, after which VGG's 4096-node classification head was replaced with the best performing head from the experiments in section 6.2.2, which was a multi-256-softmax head. This way, only the convolutional features from VGG were used in the new task, providing an insight into the efficiency of the features from VGG. Finally, VGG was trained from scratch with a multi-256-softmax head and data normalization as in the static experiments, to see if transfer learning has any effect. The networks that were pre-loaded with the VGG weights were trained with the Adam optimizer set initially to 1/10$^{\text{th}}$ the normal learning rate.

Table 3: Results showing the average precision (AP), average recall (AR) per class and mean average precision (mAP), mean average recall (mAR) and the F1 measure for several arrangements of the VGG network with and without transferring pre-trained features. The best performing network so far has been provided as comparison.

Layer Layout	pre-loaded weights	AP Author	AR Author	AP Abstract	AR Abstract	AP Title	AR Title	**mAP**	**mAR**	**F1**
VGG single-4096-softmax	yes	96.28	94.91	97.57	95.33	98.94	98.59	97.60	96.27	96.93
VGG multi-256-softmax	yes	**97.90**	**96.50**	97.68	**95.39**	98.84	98.67	98.14	**96.85**	**97.49**
VGG multi-256-softmax	no	96.81	96.21	**98.11**	95.30	**99.00**	98.92	**98.17**	96.81	97.48
2-3-3-3 multi-256-softmax	no	96.85	95.56	97.17	95.27	98.97	**99.09**	97.66	96.64	97.15

From the results in table 3, it becomes apparent that firstly, adjusting VGG by appending a multi-256-softmax head gives better performance than VGG with the original head and weights loaded until the final fully-connected layer. Secondly, transferring the features from VGG does not give the network any better performance in the new task domain, since this network achieves almost exactly the same F1 performance when the pre-trained weights are loaded as when the network is trained from scratch. This shows that the network is learning features that are inherently different from the image classification domain. Finally, the deeper VGG network has a better performance than the custom networks that were trained so far, adding to the the result from section 6.2.1, showing that more layers lead to better performance.

6.2.4 Resolution of the input image

The networks, so far, were trained on 224x224 images. At this resolution, the network is likely to depend almost solely on layout based features, as almost all of the text in the image is not legible. Also, the amount of spatial information that is retained in the last convolutional layer of the network is fairly limited, as this feature map is only 7x7 spatially (512 in depth). The network might be able to benefit from being able to distinguish words or having more spatial information in the final feature map. To test whether this holds true, the best performing network so far was trained on images with a resolution of 380x500 pixels, which is just enough to be able to read most of the text. This is also the standard size of images for PASCAL-VOC 2007, which will be discussed a bit more in section 6.3. Figure 18 in the Appendix shows the difference between the two resolutions. The best performing network so far was VGG-16 with pre-trained features and a multi-256-softmax head, so this network was trained on bigger images to see if it would perform better. Note that this increase in resolution increased the training time by a factor 4, as the input to the network increases by a factor 4. The results show that an increase in resolution does not lead to an increase in performance. This shows that the network does not depend on fine-grained spatial information or on legible words in the input image.

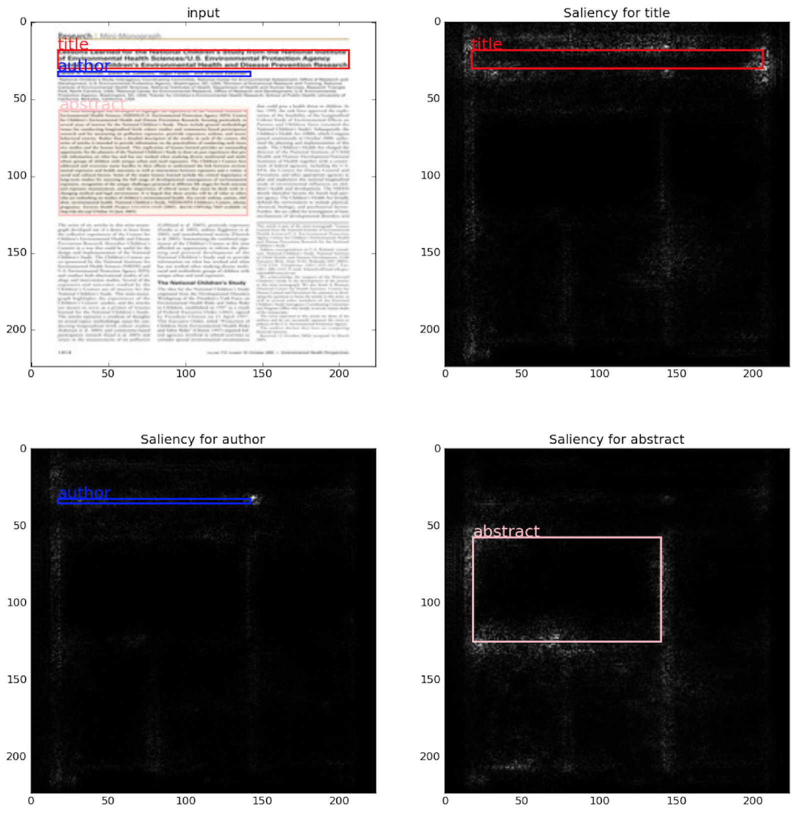
Table 4: Results showing the effect of scaling up the resolution for VGG-16 with a multi-256-softmax head.

Resolution	AP Author	AR Author	AP Abstract	AR Abstract	AP Title	AR Title	**mAP**	**mAR**	**F1**
224x224 pixels	**97.90**	**96.50**	97.68	**95.39**	**98.84**	**98.67**	**98.14**	**96.85**	**97.49**
380x500 pixels	97.48	95.71	**97.83**	94.91	98.50	98.34	97.94	96.32	97.12

6.2.5 Analysis of learned features through saliency maps

It is possible to analyse the features learned by the convnet through Guided Backpropagation [26]. In short, guided backpropagation is a method with which we can compute an imputed version of the gradient back through the network from the class node in the last classification layer. This method augments the pixels in the input image that led the network to classify the image as belonging to a certain class. These mappings are also called *saliency maps*. In order to determine which areas of the image caused the network to output the bounding box coordinates, we can use guided backprop to compute the gradient against all four of the classified coordinates of the bounding box. An example of guided backprop is shown in figure 14. By looking at the saliency maps, it becomes apparent that the network has mostly learned to respond to layout features like alignment, line spacing and line-height. It does not seem to look at the content of any of the bounding boxes. This makes it probable that the network is learning how to recognise certain templates of documents, rather than understanding what the entity truly is. This would also explain why scaling up the image resolution does not increase performance, as all of the template information is already available at lower resolution.

Figure 14: Saliency map showing the absolute saliency per bounding box per class.



6.3 Dynamic classes - Faster R-CNN

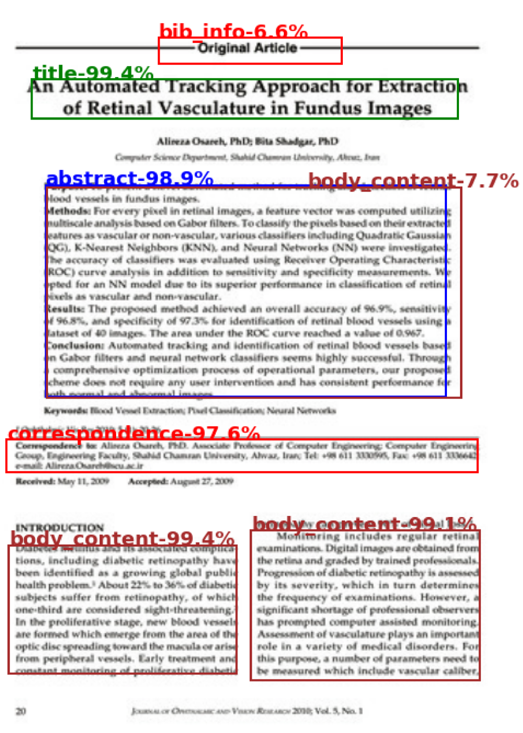
To classify dynamic classes, the Faster R-CNN framework was tested under two different settings, to give some exploratory findings about the potential for using this framework in the context of NER on document images. Faster R-CNN is a complex framework that has many different hyperparameters that might be tweaked to increase performance, but this is beyond the scope of this work. The Python implementation of Faster R-CNN that is available on GitHub [2] was used in the experiments. Both experiments trained the network using the approximate joint training method, which is explained in detail in slides provided by R. Girshick [9]. The same train-validation-test split of 80%, 10% and 10% was used for these experiments. The first experiment trained Faster R-CNN with the default VGG-network and default settings on all the classes in the document, which are 22 classes in total. As mentioned before, the joint training procedure of the convnet together with an RPN is different from normal fine-tuning of a convnet to a new domain. As a result, VGG might not be able to adapt properly to the new domain. To test this theory, VGG was replaced by the AlexNet that was pre-trained by Harley et al. [11] to classify 400,000 documents in 16 different classes. The same training procedure was followed as for the VGG experiment.

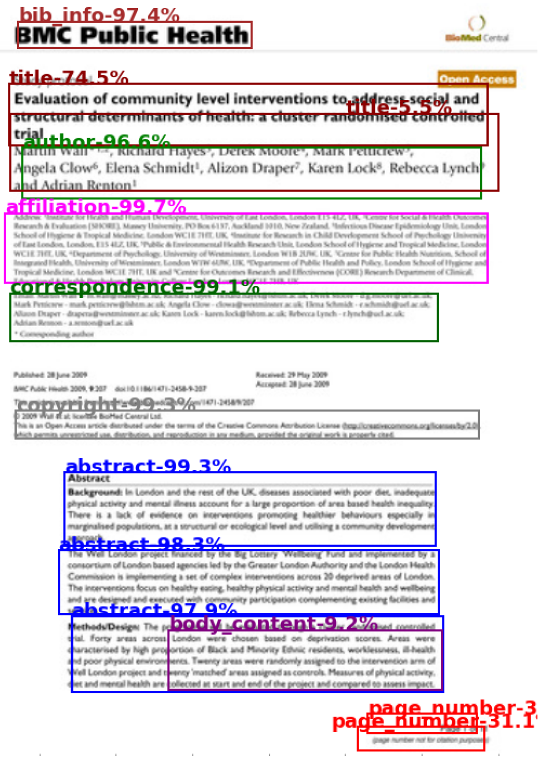
6.3.1 Interpreting the Faster R-CNN predictions

Faster R-CNN uses the PASCAL-VOC 07 metric to evaluate its predictions. This metric measures the intersection over union (IoU) between the predicted bounding boxes and the ground truth bounding boxes, counting a prediction correct if it overlaps at least 50%. However, the data in the dataset is often fragemented inconsistently, so the IoU measure between the predictions made by the network and those in the ground truth are not representative of the true performance of the network, as a prediction can encapsulate a zone correctly that is fragmented into multiple zones in the ground truth, and thus will be classified incorrectly due to a low IoU with any of the fragmented zones. Alternatively, the results can be interpreted in the same way as for the experiments with static classes, where the amount of correctly classified words per class is taken as a measure. The predictions of Faster R-CNN are bounding box coordinates with confidence measures, as illustrated in figure 15. A word can then be classified by assigning it the class label of the bounding box with the highest confidence value with which that word overlaps.

Table 5 shows the results for both the experiments with Faster R-CNN. For the full table of results, showing performance over all classes, the reader is referred to table 6 in the appendix. The results on title and abstract are comparable to the results obtained by the best network for static classes, albeit slightly worse (~-3% F1). The network that was based on VGG has a better overall performance than the network with the AlexNet pre-trained on document classification. The results show that both networks perform better on classes that have bigger entities, like body_content, abstract and title. Classes like author and dates, that are often printed on a single line, have a much lower precision and recall. This suggests that some entities are too small for the network to recognise and train on properly, which

Figure 15: Predictions from Faster R-CNN with VGG with class label and confidence value. Words that are contained within overlapping bounding boxes get labeled to the class with the highest confidence interval. This ensures that the abstract and right author are labeled correctly. In both cases, the network misses some classes.





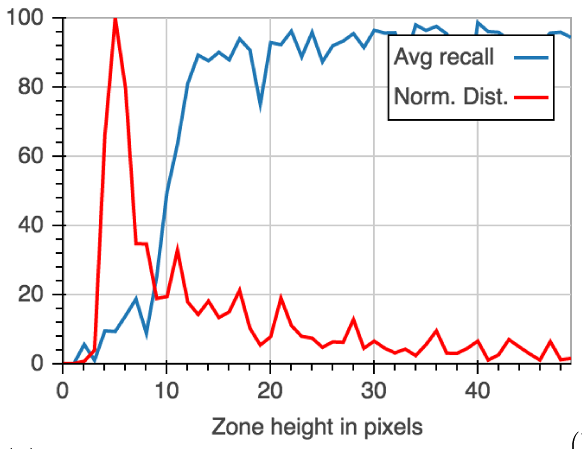
would explain the poor performance on the author class. This becomes apparent when plotting the average recall of all entities in the dataset against their height, in pixels. This is done in figure 16a, which simultaneously shows the normalized probability distribution of the height of an entity, showing that a large part of the zones in the dataset have a zone height of 15 pixels or smaller, with a peak of 5 pixels.

Figure 16b, showing the height distributions of zones both the title and the author class, confirms this theory that smaller are recognised significantly worse, as it is clear that for the author class the majority of entities has a pixel height of 8 pixels or less. For VGG, the receptive field of the first stack of convolutional layers is 5x5. These layers contain very low-level filters that detect edges and colors, and it is likely that the network can therefore not differentiate between different classes of these sizes.

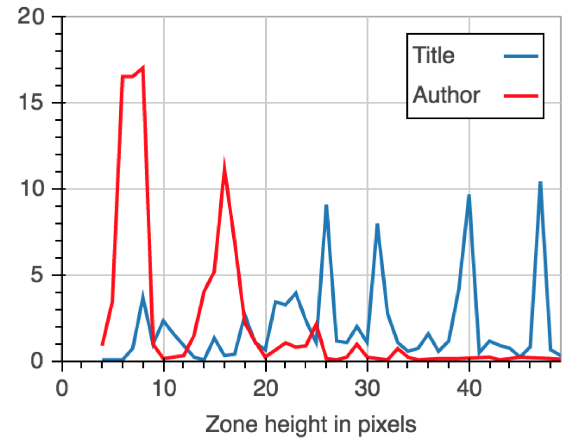
Table 5: Results for the R-CNN experiments based on VGG and AlexNet. Results are shown for the same classes as the static experiments as well as globally, across all 22 classes.

	VGG-16			AlexNet			
class	precision	recall	F1	precision	recall	F1	occurrences
abstract	93.33	92.88	**93.10**	90.20	83.60	86.90	1215
affiliation	77.54	76.81	**77.17**	60.28	58.44	59.36	1286
author	49.07	49.22	49.15	53.58	54.10	**53.84**	1163
bib_info	77.64	59.51	**68.57**	44.10	30.69	37.39	1287
body_content	94.38	92.91	**93.64**	60.58	26.91	43.75	1192
dates	16.48	15.00	**15.74**	9.92	10.09	10.00	784
editor	00.00	00.00	00.00	00.00	00.00	00.00	231
...	...	...	...	...	...	...	...
title	91.73	93.82	92.78	93.56	95.98	**94.77**	1147
type	14.70	14.64	**14.67**	1.13	0.58	0.85	516
mean	44.77	42.44	**43.61**	23.77	21.09	22.42	

Figure 16: Analysis of zone height versus performance in Faster R-CNN. The y-scale is in percentages.



(a) Average recall versus entity height in pixels and normalized distribution of entity height.



(b) Height distribution of title versus author. The network performs much better on the title class than the author class.

Ideally, Guided Backpropagation [26] would also be used to analyse the pixels in the input image by which Faster R-CNN generates its predictions, as was done for the static networks. However, because of the complexity of the Faster R-CNN framework, the implementation of Guided Backpropagation with regards to Faster R-CNN is beyond the scope of this work, and assumptions about the features used by Faster R-CNN are made by looking at the predictions generated by the network, as will be discussed in detail in the next section.

7 Discussion

The networks were trained on a random split of all the data, where no distinction has been made between publisher templates. This makes it plausible that every publisher template is contained multiple times in both the training and test set. Therefore, both could have overfit the data template and styling-wise, and might not perform well on templates that are not in the dataset. For the custom networks, this is probably more so, since the networks have been shown to respond only to template-like features. To test this, the best performing static network (VGG-16 pre-loaded weights, multi-256-softmax head) and Faster R-CNN with VGG have been applied to all the references of this work, which are guaranteed not to be in the dataset and are expected to have templates not contained in there, because of the difference in research field. A visual comparison is given in figure 17 in the appendix. The comparison confirms that the networks have overfit on the GROTOAP2 dataset, as both networks perform well on only a few of the documents. The difference in approaches becomes even more clear. The static network completely misses some entities or assigns them incorrectly. On the other hand, Faster R-CNN seems to recognise almost all of the entities in the network correctly, but confuses them class-wise. For instance, Faster R-CNN seems to associate narrow blocks of text with body_content, while it classifies the wider blocks of text as abstract. This can be explained by looking at the dataset, where the abstract is mostly wide, bold or italic, whereas in the dataset it is often slim and without styling. For the static networks, it is clear that it uses alignment features to classify its entities, which can be seen by the network predicting the abstract incorrectly across columns several times. Also, that it performs poorly for unseen templates becomes clear by looking at the author classification, as some papers have the authors in three separate boxes, which is unseen in GROTOAP2. The static network completely misses these entities, while Faster R-CNN, which has been trained on single author boxes, is able to recognise all of them. This implies that Faster R-CNN is able to recognise these entities based on local features, rather than global features.

From the experiments, their results and the visual comparison of both networks' predictions on a new set of documents, the difference between the two methods of NER becomes clear. When training a network on the full image, the network tends to learn how to recognise templates, rather then the entities itself. With Faster R-CNN, on the other hand, the network is trained on on image crops and region proposals are given based on a small window of features from the final feature map. This means that the network does not have the full image available in order to classify the entities in the document and the network is forced to recognise entities in other ways than template learning. This becomes clear when looking at the predictions made by Faster R-CNN, as the network is able to recognise (but sometimes wrongly classify) all of the zones in the document, even though the template is not in the training set. Overall, the performance of Faster R-CNN on the bigger entities in the document is comparable to the networks trained on static classes. Also, Faster R-CNN was able to achieve this performance on a more varied set of documents, as the network was not limited to documents that have only one author and title, as was the case for the other networks. Whether Faster R-CNN can achieve the same performance on smaller entities as well, is something for future research.

8 Conclusions

This work has been a study into performing NER on images of documents by means of convolutional neural networks. The research questions have been answered as follows, in occurring order. Firstly, a good architecture for performing the NER task statically has been found to be a deep neural network like VGG-16, adjusted by appending multiple small softmax classification heads, one head for each entity. The VGG-16 based network has been shown to achieve an average F1 performance of 97.49% on several entities in templates from over 200 different publishers. Secondly, transferring learned features from an image classification task has been shown to have a limited effect, as the network achieves nearly identical performance when it is trained from scratch without transfer learning. Thirdly, the Faster R-CNN networks have been shown able to perform the NER task dynamically, with comparable performance as the static networks. Fourthly, the static networks have been shown likely to respond to global template-like features, such as alignment and spacing. This was done through experiments involving image resolution and by generating saliency maps. By looking at the average recall versus the size of the entities and by comparing the performance of both methods on a new set of documents, it has been shown likely that Faster R-CNN responds to more local features of the textual entities, like shape and styling. Both methods have their strong and weak points. When the NER task is to extract a predefined set of entities from a limited amount of different templates, then the static approach will likely be able to achieve the best performance. However, analysis suggests that this approach will not generalize well to templates not contained in the training set, so having a training set that contains a complete set of templates is paramount to achieving good performance. Faster R-CNN is more suited for cases where the amount of entities in a document can vary greatly, as the network classifies entities dynamically and has been shown likely to respond to local features in the document, rather then the template. While Faster R-CNN generalises better, it still has a high dependency on the training set in terms of the styling of the entities. Also, Faster R-CNN performs worse on small entities in the document, but its performance is expected to improve when the algorithm and the settings of the training procedure are fine-tuned towards the specific NER task.

Concluding, this work has made the following contributions. Firstly, to the author's knowledge, it has been the first work to use convolutional neural networks to do NER solely on the pixel representation of documents, providing exploratory insights into the applicability of such networks in automated NER systems. Secondly, it has provided an overview of how different network sizes, architectures and image resolutions influence NER by means of convnets on documents and has shown the influence of transfer learning with respect to image classification networks (VGG-16). Finally, it has given an initial overview into using the dynamic object detection framework Faster R-CNN for entity recognition and has provided a comparison between both static and dynamic approaches in terms of the learned features and the performance on an NER task.

9 Future work

The experiments in this work have been performed on a single random train-validation-test split of the data, and results might differ when the networks are trained on a different split. Therefore, in order to achieve fully reliable results, the experiments should be cross-validated across many different splits, but due to time limitation this was not possible (one experiments runs 12 ± 3 hours). Additionally, for the static networks, the only deep convolutional neural network that has been experimented with was VGG-16. A more advanced network such a GoogLeNet [27] or ResNet [12] might be able to achieve better performance on the NER task. Furthermore, the Faster R-CNN framework could be adjusted in order to improve performance. By scaling up the resolution such that most of the entities are at least 10 pixels high, the network might be able to learn how to distinguish between the smaller entities in the dataset, massively improving performance on some of the entities classes. However, this might also reduce performance on the bigger entities, as the receptive field of the network will not be big enough to encapsulate larger entities like title. Another way of handling this problem would be to also fine-tune the first few layers of the VGG-16 network, as this is not the Faster R-CNN default. Also, in this work there has been no hyperparameter tuning for Faster R-CNN, and a thorough search through the hyperparameter space could improve performance. Finally, this work has researched NER with regards to scientific documents. Running the experiments on a different dataset of for instance receipts or invoices would provide valuable insight into the practical application value of these technologies in this work.

Appendix

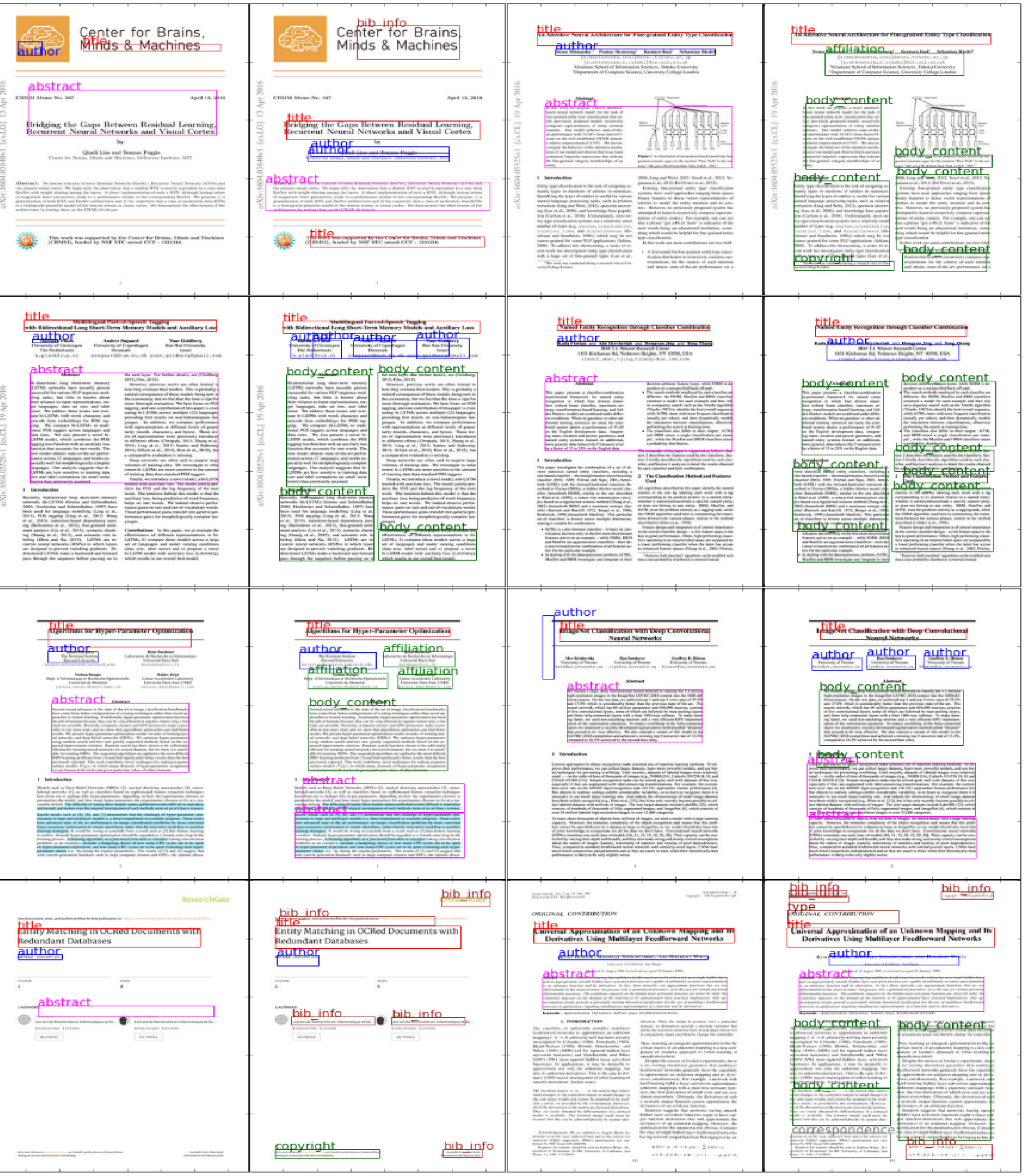
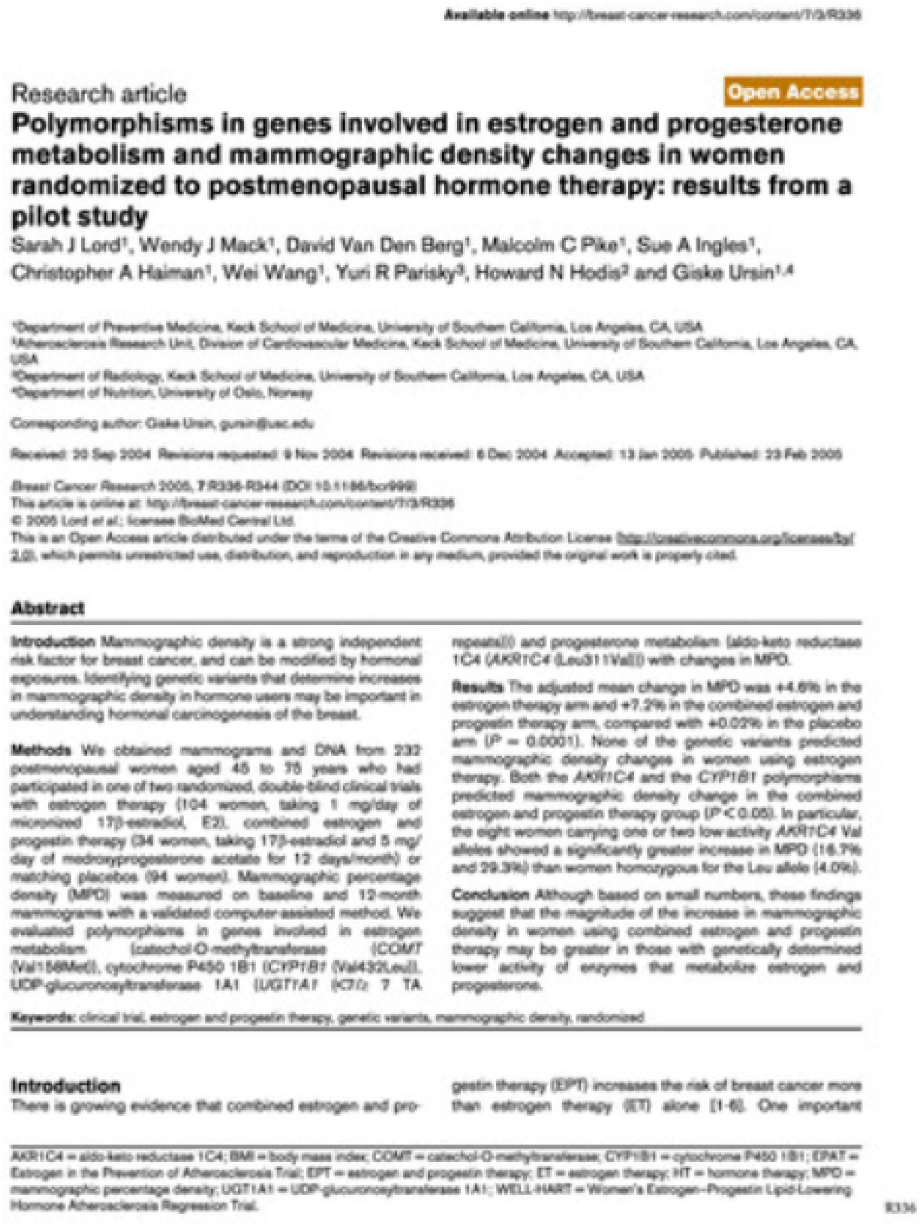


Figure 17: Comparing of the output of VGG-16 with a multi-256-softmax and Faster R-CNN. The first and third column are the results from VGG-16 and the second and fourth column are Faster R-CNN predictions. The Faster R-CNN predictions only show the prediction with the highest confidence for a given region.

Figure 18: A comparison of resolutions.

(a) A 224x224 pixel image



Available online http://breast-cancer-research.com/content/7/3/R336

Research article

Open Access

Polymorphisms in genes involved in estrogen and progesterone metabolism and mammographic density changes in women randomized to postmenopausal hormone therapy: results from a pilot study

Sarah J Lord[1], Wendy J Mack[1], David Van Den Berg[1], Malcolm C Pike[1], Sue A Ingles[1], Christopher A Haiman[1], Wei Wang[1], Yuri R Parisky[3], Howard N Hodis[2] and Giske Ursin[1,4]

[1]Department of Preventive Medicine, Keck School of Medicine, University of Southern California, Los Angeles, CA, USA
[2]Atherosclerosis Research Unit, Division of Cardiovascular Medicine, Keck School of Medicine, University of Southern California, Los Angeles, CA, USA
[3]Department of Radiology, Keck School of Medicine, University of Southern California, Los Angeles, CA, USA
[4]Department of Nutrition, University of Oslo, Norway

Corresponding author: Giske Ursin, gursin@usc.edu

Received: 20 Sep 2004 Revisions requested: 9 Nov 2004 Revisions received: 6 Dec 2004 Accepted: 13 Jan 2005 Published: 23 Feb 2005

Breast Cancer Research 2005, **7**:R336-R344 (DOI 10.1186/bcr999)
This article is online at: http://breast-cancer-research.com/content/7/3/R336
© 2005 Lord *et al.*; licensee BioMed Central Ltd.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/2.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

Introduction Mammographic density is a strong independent risk factor for breast cancer, and can be modified by hormonal exposures. Identifying genetic variants that determine increases in mammographic density in hormone users may be important in understanding hormonal carcinogenesis of the breast.

Methods We obtained mammograms and DNA from 232 postmenopausal women aged 45 to 75 years who had participated in one of two randomized, double-blind clinical trials with estrogen therapy (104 women, taking 1 mg/day of micronized 17β-estradiol, E2), combined estrogen and progestin therapy (34 women, taking 17β-estradiol and 5 mg/day of medroxyprogesterone acetate for 12 days/month) or matching placebos (94 women). Mammographic percentage density (MPD) was measured on baseline and 12-month mammograms with a validated computer-assisted method. We evaluated polymorphisms in genes involved in estrogen metabolism [catechol-O-methyltransferase (*COMT* (Val158Met)), cytochrome P450 1B1 (*CYP1B1* (Val432Leu)), UDP-glucuronosyltransferase 1A1 (*UGT1A1* (*CTA*) 7 TA repeats))] and progesterone metabolism (aldo-keto reductase 1C4 (*AKR1C4* (Leu311Val))) with changes in MPD.

Results The adjusted mean change in MPD was +4.6% in the estrogen therapy arm and +7.2% in the combined estrogen and progestin therapy arm, compared with +0.02% in the placebo arm (*P* = 0.0001). None of the genetic variants predicted mammographic density changes in women using estrogen therapy. Both the *AKR1C4* and the *CYP1B1* polymorphisms predicted mammographic density change in the combined estrogen and progestin therapy group (*P* < 0.05). In particular, the eight women carrying one or two low-activity *AKR1C4* Val alleles showed a significantly greater increase in MPD (16.7% and 29.3%) than women homozygous for the Leu allele (4.0%).

Conclusion Although based on small numbers, these findings suggest that the magnitude of the increase in mammographic density in women using combined estrogen and progestin therapy may be greater in those with genetically determined lower activity of enzymes that metabolize estrogen and progesterone.

Keywords: clinical trial, estrogen and progestin therapy, genetic variants, mammographic density, randomized

Introduction

There is growing evidence that combined estrogen and progestin therapy (EPT) increases the risk of breast cancer more than estrogen therapy (ET) alone [1-6]. One important

AKR1C4 = aldo-keto reductase 1C4; BMI = body mass index; COMT = catechol-O-methyltransferase; CYP1B1 = cytochrome P450 1B1; EPAT = Estrogen in the Prevention of Atherosclerosis Trial; EPT = estrogen and progestin therapy; ET = estrogen therapy; HT = hormone therapy; MPD = mammographic percentage density; UGT1A1 = UDP-glucuronosyltransferase 1A1; WELL-HART = Women's Estrogen–Progestin Lipid-Lowering Hormone Atherosclerosis Regression Trial.

R336

(b) A 380x500 pixel image

Table 6: Full table of results for the Faster R-CNN with VGG versus AlexNet.

	VGG-16			AlexNet			
class	precision	recall	F1	precision	recall	F1	occurrences
abstract	93.33	92.88	**93.10**	90.20	83.60	86.90	1215
acknowledgment	00.00	00.00	00.00	00.00	00.00	00.00	0
affiliation	77.54	76.81	**77.17**	60.28	58.44	59.36	1286
author	49.07	49.22	49.15	53.58	54.10	**53.84**	1163
bib_info	77.64	59.51	**68.57**	44.10	30.69	37.39	1287
body	00.00	00.00	00.00	00.00	00.00	00.00	0
body_content	94.38	92.91	**93.64**	60.58	26.91	43.75	1192
conflict_statement	26.16	22.88	**24.52**	00.00	00.00	00.00	269
copyright	65.88	63.35	**64.61**	26.43	25.43	25.93	982
correspondence	47.82	49.47	**48.64**	27.36	28.01	27.69	816
dates	16.48	15.0	**15.74**	9.92	10.09	10.0	784
editor	00.00	00.00	00.00	00.00	00.00	00.00	231
equation	00.00	00.00	00.00	00.00	00.00	00.00	1
figure	00.00	00.00	00.00	00.00	00.00	00.00	12
glossary	39.79	41.23	**40.51**	00.00	00.00	00.00	32
keywords	39.82	38.73	**39.27**	7.33	6.36	6.85	382
page_number	18.67	19.13	**18.90**	00.00	00.00	00.00	695
references	76.13	75.4	**75.77**	00.00	00.00	00.00	51
table	56.68	42.4	**49.54**	0.02	00.00	00.00	44
title	91.73	93.82	92.78	93.56	95.98	**94.77**	1147
title_author	00.00	00.00	00.00	00.00	00.00	00.00	0
type	14.7	14.64	**14.67**	1.13	0.58	0.85	516
unknown	5.89	2.76	4.33	1.22	0.75	0.98	377
mean	44.77	42.44	**43.61**	23.77	21.09	22.42	

References

[1] Neural networks. http://ufldl.stanford.edu/wiki/index.php/Neural_Networks.

[2] py-faster-rcnn. python implementation of faster r-cnn. https://github.com/rbgirshick/py-faster-rcnn.

[3] K. Chellapilla, S. Puri, and P. Simard. High performance convolutional neural networks for document processing. In *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft, 2006.

[4] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[5] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics, 2003.

[6] M. Ford. *Rise of the Robots: Technology and the Threat of a Jobless Future.* Basic Books, 2015.

[7] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[8] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[9] R. Girshick. Training r-cnns of various velocities. `https://www.dropbox.com/s/xtr4yd4i5e0vw8g/iccv15_tutorial_training_rbg.pdf`, 2015.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[11] A. W. Harley, A. Ufkes, and K. G. Derpanis. Evaluation of deep convolutional nets for document image classification and retrieval. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 991–995. IEEE, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[13] R. Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 593–605. IEEE, 1989.

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[15] J. J. Hopfield. Artificial neural networks. *Circuits and Devices Magazine, IEEE*, 4(5):3–10, 1988.

[16] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, 3(5):551–560, 1990.

[17] Z. Huang, W. Xu, and K. Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[20] A. Mikheev, C. Grover, and M. Moens. Description of the ltg system used for muc-7. In *Proceedings of 7th Message Understanding Conference (MUC-7)*, pages 1–12. Fairfax, VA, 1998.

[21] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[22] C. Poultney, S. Chopra, Y. L. Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2006.

[23] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[24] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[28] S. L. Taylor, M. Lipshutz, and R. W. Nilson. Classification and functional decomposition of business documents. In *icdar*, page 563. IEEE, 1995.

[29] D. Tkaczyk, P. Szostek, and L. Bolikowski. Grotoap2 the methodology of creating a large ground truth dataset of scientific articles. *D-Lib Magazine*, 20(11):13, 2014.

[30] J. Weng, N. Ahuja, and T. S. Huang. Cresceptron: a self-organizing neural network which grows adaptively. In *Neural Networks, 1992. IJCNN., International Joint Conference on*, volume 1, pages 576–581. IEEE, 1992.

[31] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer vision–ECCV 2014*, pages 818–833. Springer, 2014.

[32] X. Zhang and Y. LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

[33] G. Zhu, T. J. Bethea, and V. Krishna. Extracting relevant named entities for automated expense reimbursement. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1004–1012. ACM, 2007.