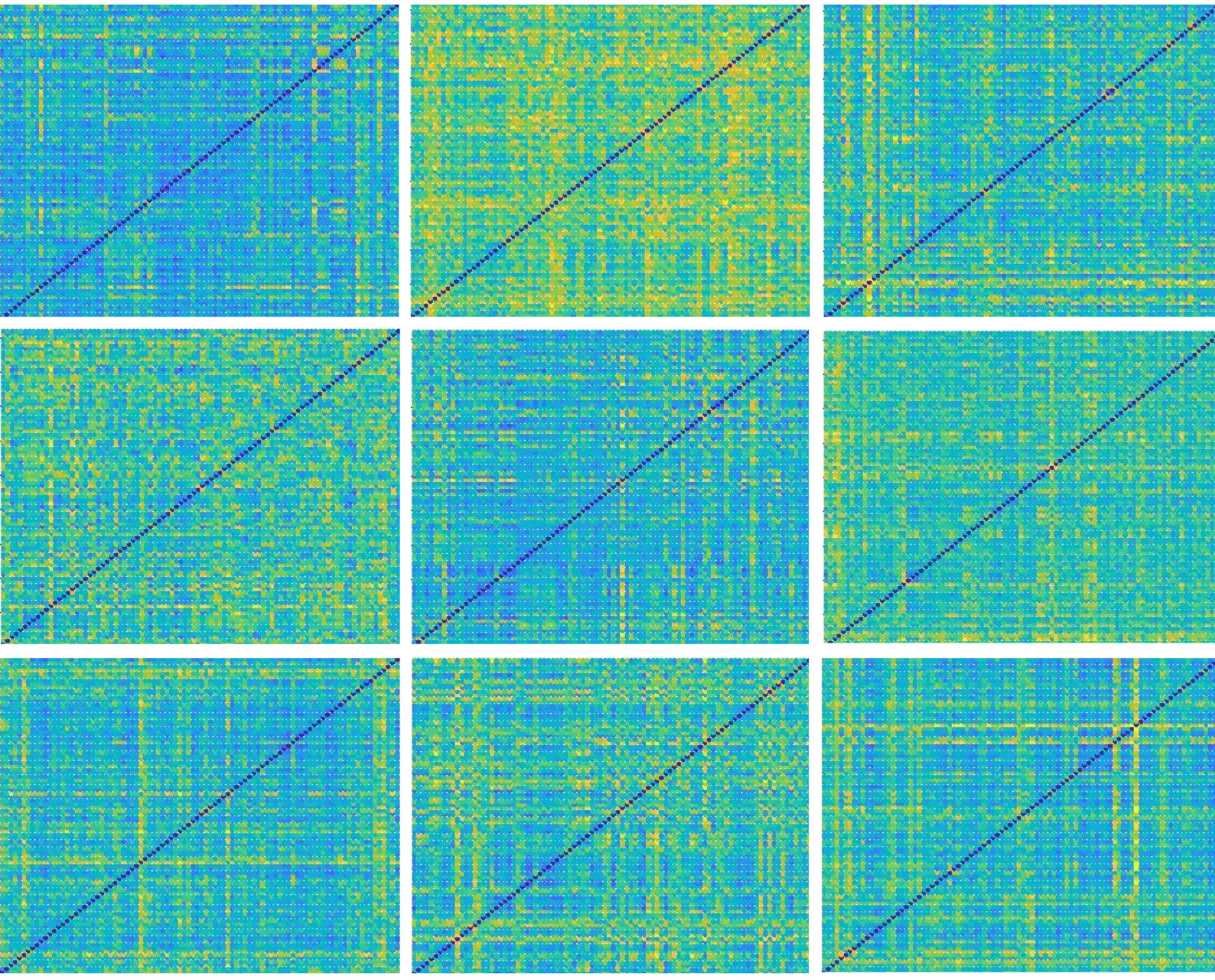
Simulating the human visual brain using deep neural networks



Anne-Ruth José Meijer

Front page image is a selection of representational similarity matrices of humans

Simulating the human visual brain using deep neural networks

Anne-Ruth José Meijer
10978046

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
dhr. dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

June 28th, 2019

Abstract

Understanding the nature of human intelligence is a topic that long fascinated neuroscientists and computer scientists. A fraction of human intelligence is recognizing objects, which is part of the visual hierarchy. Understanding how this process works in a human brain could benefit object-recognition in networks, allowing the development of more advanced neural networks. This research is based on a challenge of the Algonauts Project, which is on the quest of understanding human intelligence. The challenge is set to let participants make a model that simulates the early visual and inferior-temporal cortex of the brain. The challenge facilitates a scoring method for networks to get an understanding of how well a network simulates the human visual brain. In this research multiple existing convolutional neural networks are compared. Based on the comparisons a new network is designed for this thesis, which is a variation of a residual neural network. This new trained network resulted in a average score of 15.53% on the challenge. Which improved on the score of 10.31%, scored by the best tested existing network (a DenseNet201). This result is achieved by only training the network for 10 epochs. Since the aim of the research is simulating the human visual brain, there is a measurement required to consider a network a simulating network. For this purpose the baseline of the Algonauts challenge is used, a network is required to score higher than 7.41 to be considered simulating a human visual brain. This score is achieved by multiple networks, where ResNet20 trained for 10 epochs outperforms the rest. The ResNet20 variation trained for 10 epochs is a shallow network trained with fewer epochs compared to the second best network, yet is performs over 5% better on the Algonauts challenge.

Acknowledgements

I like to express my deepest gratitude to dr. Arnoud Visser for helping me finding a suitable subject for my bachelor thesis and his support during the research. Without his help I would never published a paper about this research.

Contents

Chapter 1

Introduction

Artificial intelligence is the science of creating a machine that simulates human intelligence [9]. To understand the nature of human intelligence it is important to understand the human brain, given that this organ provides humans with intelligence. Projects like the Algonauts Project [4] and Brain-Score [14] are started to understand and simulate the human brain. Both projects rank convolutional neural networks, searching for the most brain-like network. The Brain-Score project uses three benchmarks to rank networks on their ability to simulate the human visual brain. The Algonauts Project launched a challenge for 2019 to simulate the human visual brain. The Algonauts challenge 2019 has the same goal as the Brain-Score project, but uses a different scoring method. The Algonauts challenge compares human functional magnetic resonance imaging (fMRI) data with data generated by artificial networks. Both projects are started by the Massachusetts Institute of Technology and aims to bring biological and artificial intelligence researchers together to advance both fields. By launching these projects the researchers gain a common platform where it is possible to exchange ideas. Understanding the nature of intelligence is not only interesting from a biological point of view, but also for computer scientists. By gaining knowledge of intelligence, it can be applied to techniques of artificial intelligence. Networks could be perfected with the knowledge and more advanced networks could be developed. Understanding intelligence could also gain the knowledge of improving upon that intelligence. A first step in understanding intelligence and therefore understanding the human brain, is simulating that brain. One task a human brain does is recognizing objects viewed by the eyes. The visual brain is composed of a so-called visual hierarchy [2]. This hierarchy is the subject of the 2019 Algonauts challenge. The challenge contains two tracks, where the first track is focused on predicting brain activity in two brain regions. The goal is to construct models that best predict brain activity in the early visual cortex (EVC) and inferior-temporal cortex (IT) in reaction to viewing an image. This track is the base for this research.
In the human brain the ventral stream is responsible for recognizing objects [2]. In computer vision this task is executed by convolutional neural networks (CNN's) [8]. CNN's have shown in related work that their internal representation of images have similarities with the way a human brain represents them [14]. This makes these types of networks useful for the quest to understand the nature of intelligence.

1.1 Research question

To simulate the human visual brain, the brain activity in the EVC and IT regions of the ventral stream will be predicted. The models predicting the brain activity are formed by CNN's. Thus, the question this research aims to answer is:

"Can the brain activity in the EVC and IT regions of the brain be predicted in response to viewing an image?"

To be able to answer this question, both regions of the brain need to be separately tested. This is caused by the different functionalities of both regions in the visual hierarchy [2]. The EVC occurs in a early stage of the visual hierarchy, while the IT region occurs later. The EVC affects two components of the recognition of an object: combining the information and processing that information in an early stage. The IT region is solely responsible for processing information and thus recognizing an object. Because the regions occur at different times in the hierarchy and have different tasks, it is expected that they are best simulated by different layers of a CNN. For this reason this research also aims to answer the two following sub-questions:

"Can the brain activity in the EVC region of the brain be predicted in response to viewing an image?" and "Can the brain activity in the IT region of the brain be predicted in response to viewing an image?"

When a brain is simulated by a CNN it is interesting to see what properties improve the ability of simulation. Some important differences between CNN's are the architecture of CNN's, the amount of layers, and the amount of training time. These properties all produced some additional sub-questions:

"Are deep CNN's better for simulating the human visual brain than shallow CNN's?", "Does training improve the ability of a CNN to simulate the human visual brain?", and "Which architecture is best for simulating the human visual brain?"

This thesis is structured in the following order. First the theoretical framework behind this research will be explained, this includes topics like the mentioned brain regions, CNN's, a measure of comparing CNN's against the human brain, and related work. Next the approach of the research is explained. After this the results are evaluated. The chapters after the approach discuss the results and conclude this research.

Chapter 2

Theoretical foundation

In this section some important theory for this research is explained. First the theoretic foundation for the used regions in the visual hierarchy is given. Second the structure of CNN's and useful CNN's are described. The CNN's include AlexNet and VGG, which are some older networks and used for comparing results. Other networks explained are CORnet, ResNet and DenseNet variations, these networks have proven to score well on simulating the human brain in related work [14]. Third an explanation of a representation method for comparing network and human data is given. Last the related work is described. For the explanation of the basics of CNN's the online Stanford course is used [17].

2.1 The early visual cortex and inferior-temporal cortex

The human visual brain affects multiple regions in the brain. Recognizing an object is processed through the visual hierarchy inside the human brain [2]. This hierarchy contains, among others, the EVC and IT regions of the brain. The EVC is responsible for the early stages of the visual hierarchy, while the IT functions in a later stage.
The early visual cortex consists of the V1, V2 and V3 [3]. The V1 combines the information from the left and right eye and is the first step in the visual hierarchy [2]. From the V1 there are two so-called pathways to the rest of the brain, the dorsal stream and the ventral stream. The dorsal stream includes the medial temporal areas, these are the V5 and the medial superior temporal (MST). This stream processes spatial information, meaning that this pathway processes information of where an object is and how it is placed. Information that is processed in this pathway among others include the motion of the object, position and depth. The ventral stream includes the V2, V4 and IT. This stream processes the properties of the viewed object, it is responsible for seeing what an object is. Properties that are processed here are for example colors and shapes. Both the EVC and IT region exist in the ventral stream, but unlike the IT region only a part of the EVC affects the ventral stream. The different areas of the brain are visualized in figure 2.1, in the figure the IT region is called the ITC and CIT. Starting at V1 the information is passed through the two pathways in a feed-forward direction. However in the hierarchy every stage has also access to a backward connection, except to the RGC. RGC is short for retinal ganglion cells, these are the cells retrieving information about objects. The full visual hierarchy is shown in figure 2.1

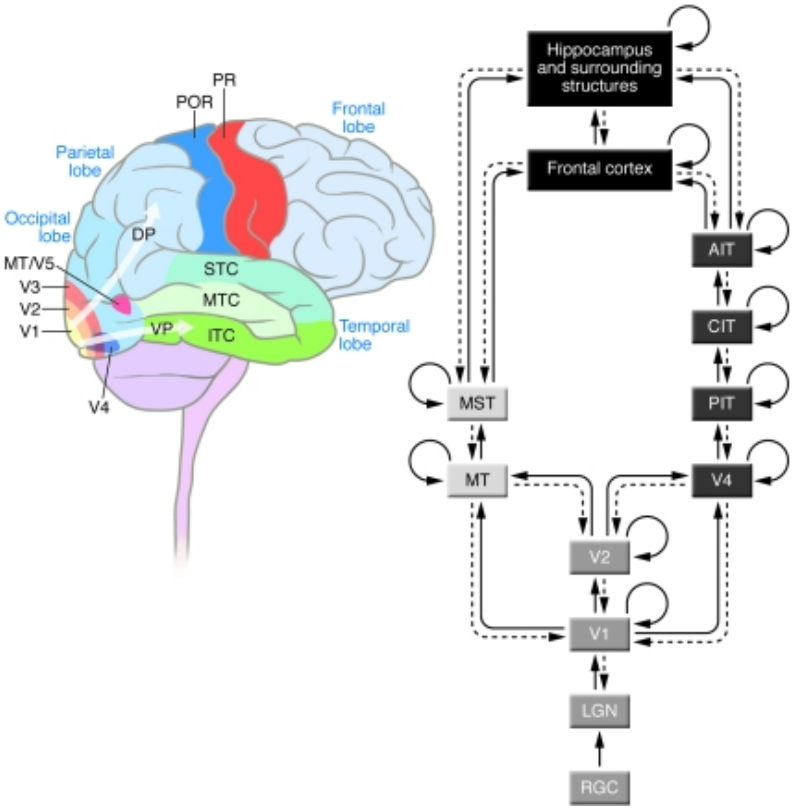


Figure 2.1: Basic architecture of the visual hierarchy [1]

2.2 Convolutional neural networks

CNN's are a type of deep neural networks that are typically used in computer vision tasks. CNN's are built up from neurons and have learnable weigths and biases, similar to shallow neural networks. The difference between the networks is that CNN's expect images as input and can process these better than other types of neural networks. Suppose a standard neural network would train on images of 100 by 100 pixels with 3 color channels. The input variables for the network would be of the size 100x100x3, meaning 30.000. With larger images the amount of input variables would grow rapidly. A standard network would learn all the weights of these input variables, which would take a lot of time and space. Instead of learning all weights separately, a CNN trains the weights of its convolutional operations. A CNN consists of a sequence of convolutional, pooling and fully-connected layers.

2.2.1 Convolutional layer

The convolutional layer is responsible for the convolutional operations within a CNN. The convolutional layer consists of a set of learnable filters. The filters have a smaller height and width size than the input image, but do have the same depth. When the input has a size of 100x100x3, a filter could have for example a size of 5x5x3. Each filter in the convolutional layer will be slid across the width and height of an image while calculating the dot product at every position. This results in a 2 dimensional activation map that shows the features a

[1] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2929717/figure/F1/

filter recognized. Every filter will activate on a different pattern. For example a filter can be trained to recognize wheel-like shapes and will activate when it recognizes this shape in an image. This will be added to the activation map of the filter and eventually will be combined with all the other activation maps. The combined activation maps result in an output for the layer. The output essentially shows the reaction of the different filters to the image and thus the different features recognized.
The size of the output of a convolutional layer is impacted by three parameters: the depth, stride and zero-padding. The depth is the parameter that corresponds to the amount of filters used to recognize patterns. This impacts the size of the output, because each activation map of each filter is combined for the output. Stride is a parameter that specifies how many pixels each filter moves to get to the next position. It essentially determines where each filter needs to stop, so it decides where every position is. When the stride is 1, the filters are moved 1 pixel at a time. If the stride is 2 the filter moves 2 pixels at a time. Meaning that a higher stride produces a smaller output. The parameter zero-padding is used for adding a border of zeroes to the input image. Zero-padding can be used to preserve the full input image in the beginning of a network, making sure no information of the image is lost. Convolutional layers can sometimes loose a part of the input of the layer. Take for example a 32x32x3 sized image where the layer applies a 5x5x3 filter with a stride of 1. In this case the spatial size of the output would be 28x28x3, causing loss of information. When the original image is padded with 2 layers of zeroes the new spatial size would be 36x36x3 and the size of the output would then be 32x32x3. The output volume of a convolutional layer can be computed by using the following formulas.

$$W_2 = \frac{W_1 - F + 2P}{S + 1} \qquad H_2 = \frac{H_1 - F + 2P}{S + 1} \qquad D_2 = K$$

In these formulas W_2, H_2 and D_2 stand for the width, height and dept of the output of the layer. W_1, H_1 and D_1 represent the width, height and depth of the input of the layer. K is the number of filters and F their spatial extend. S is the stride of the layer and P the zero-padding.
After each type of convolution operation a ReLu layer is used to ensure non-linearity. The ReLu layer applies a ReLu function to the activation maps produced by the convolutional layer. A ReLu function could be for example $max(0, x)$, ensuring that there are no values below zero. The ReLu layer does not impact the spatial size of the output.

2.2.2 Pooling layer

Pooling layers are commonly used in between consecutive convolutional layers. Pooling layers are used for reducing the spatial size of the input for the next layer. By using pooling layers the amount of parameters passed through will be reduced. With this method the amount of computations is also reduced, preventing overfitting.
Overfitting is a phenomenom where a network is only good at predicting the trained data set. With overfitting the network is adapted to the parameters of one data set, causing it to be bad at making predictions about never-seen data. Overfitting causes the network to link certain patterns to data, while in reality these patterns are not caused by that data.
An example of a pooling layer is the *max pooling* layer. At each location within a max pooling layer the layer chooses the maximum value in the given window. An example of a max pooling operation is visualized in figure 2.2. Here a pooling window of 2x2 with stride 2 is used to slide over the input of the layer. At every location the largest number is chosen to put in the output. This type of pooling window discards 75% of the input. Another type of pooling layer is *average pooling*. Instead of choosing the highest number inside the pooling window, average pooling takes the average of all numbers inside the window. A pooling layer only affects the width and height of the input image, the depth remains unchanged.

Formulas used to calculate the spatial size of the output of the pooling layer are the following:

$$W_2 = \frac{W_1 - F}{S + 1} \qquad H_2 = \frac{H_1 - F}{S + 1} \qquad D_2 = D_1$$

In these formulas the W_1, H_1, D_1, W_2, H_2 and D_2 have the same meaning as the previous shown formulas. F represents the spatial size and S the stride of the pooling window. Typically a pooling window is not bigger than a 3x3 window, because otherwise it would become too destructive and the network would lose too much information.

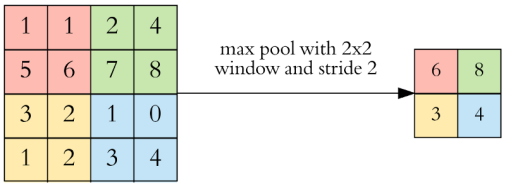


Figure 2.2: A max pooling operation[2]

2.2.3 Fully-connected layer

A CNN typically finishes with a fully-connected layer. This is the same type of fully-connected layer as in other neural networks. In these layers the different features of the input are combined into a vector. This layer essentially combines the features to classify the input. For example features like round shapes, and black and white hexagons inside the round shapes are put together to find the classification of a football.

2.2.4 Dropout layer

The dropout layer is another way to prevent overfitting in a CNN. The layer essentially drops some random activations and sets them to zero. It forces the network to classify images even when activations are dropped. Causing the network to not be dependent on every aspect of the data and not be too fitted to the data.

2.2.5 AlexNet

AlexNet was developed in 2012 and ranked first on recognizing images in the ImageNet ILSVRC challenge[3] by a large margin [11]. This challenge evaluates networks for image classification at a large scale. The AlexNet consists of five convolutional layers, with ReLu after each convolution, and three fully-connected layers. Prior to the first and second fully-connected layer a dropout layer is placed [11]. The full architecture is visualized in figure 2.3. The network takes input of the size 227x227 pixels, including an image of the size 224x224 with a padding of 2. In the first convolutional layer the kernels have the size of 11x11x3, in later layers the size decreases to 5x5x3 and 3x3x3. The last fully-connected layer sends his output to a softmax operation which is a classifier with 1000 class labels.

[2]https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

[3]http://image-net.org/challenges/LSVRC/2016/index

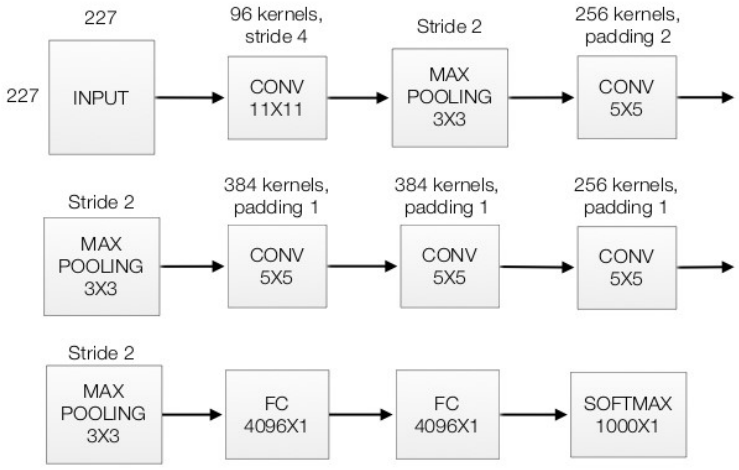


Figure 2.3: AlexNet architecture[4]

2.2.6 VGG

The VGG network is developed in 2014, the network showed that depth does make a difference for a good performance [15]. In 2014 it was ranked second for classifying images by the ImageNet challenge. The depth of a VGG can vary from 11 layers up to 19 layers, this research uses a 19 layer variation. In figure 2.4 the architecture of the VGG with 19 layers is shown. The layers include multiple convolutional, max pooling and fully-connected layers. The amount of convolutions is dependent on the amount of layers in the VGG. However what does not vary is the size of the convolution and max pooling kernels, the convolution kernels have a size of 3x3x3 and the max pooling kernels a size of 2x2x2. Every VGG type includes five max pooling layers, three fully-connected layers and a softmax layer. Before the first two fully-connected layers a dropout layer occurs. The input of the network is a 224x224 sized image, not including padding like the AlexNet input.

[4]Image made by Anne-Ruth Meijer

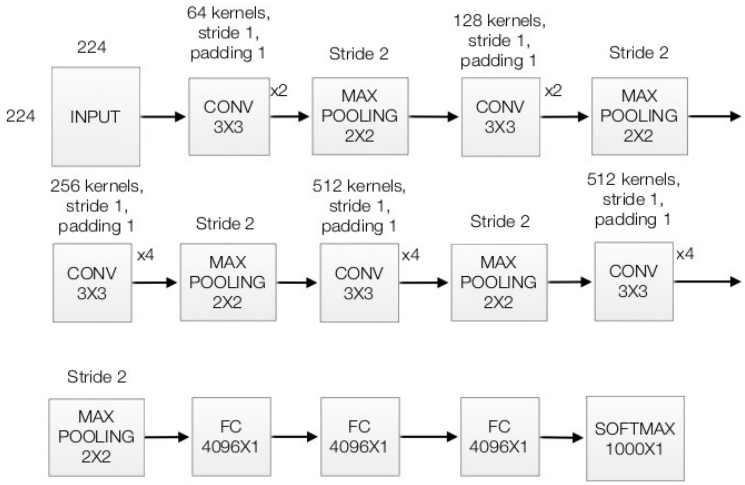


Figure 2.4: VGG 19 layer architecture[5]

2.2.7 ResNet

A ResNet stands for a Residual neural network [6]. ResNets were developed to resolve the problem of a convolutional neural network that goes "too deep". Often with deep neural networks the assumption is made that deeper is better, however after some depth the performance can degrade rapidly. This can for example occur in a network like VGG, it is caused by the vanishing gradient problem.
The vanishing gradient problem can occur when a network is formed with a lot of layers. When the network backpropagates and the gradient of loss function gets smaller, the weights can be prevented from updating. As a result the earlier layers train slower and can even return inaccurate results. The small gradient means that the earlier layers in a network will not be accurately trained and can even halt the full training process. This decreases the accuracy of the whole network, because the earlier layers in a network are especially important for a neural network. The earlier layers are responsible for detecting the simpler patterns in an image, for example edges. These simple patterns are often crucial for recognizing an object. A ResNet avoids the vanishing gradient problem by possessing the possibility of skipping layers. This means that the gradient can directly flow from initial layers to later layers.

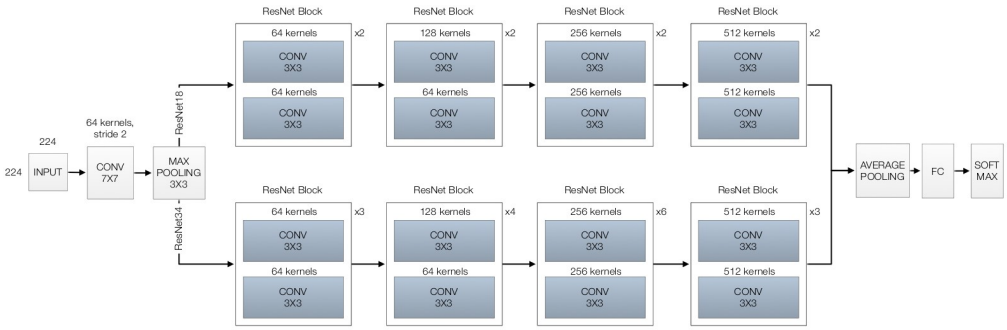


Figure 2.5: ResNet 18 and 34 layered architectures[6]

[5]Image made by Anne-Ruth Meijer

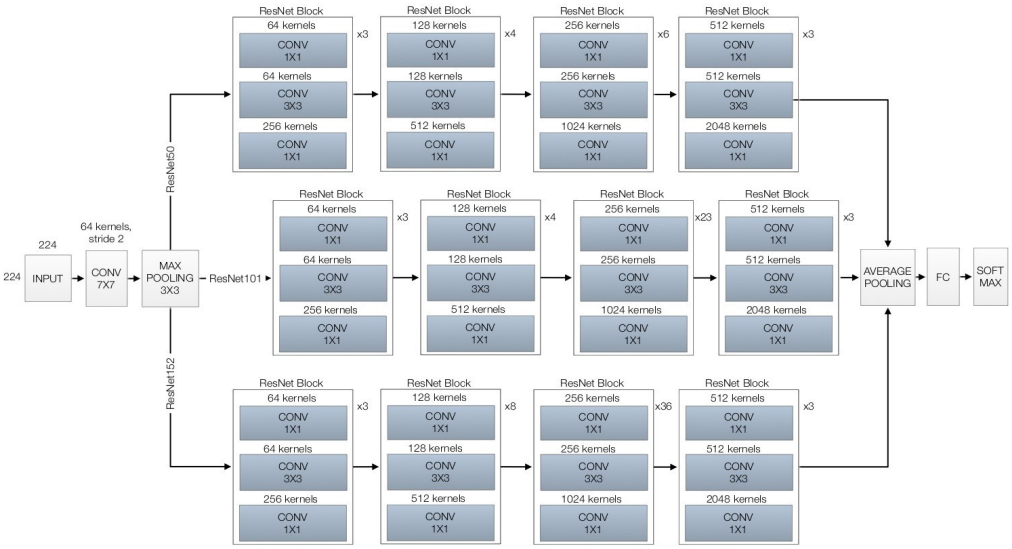
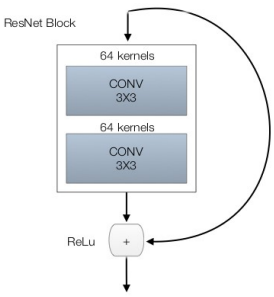


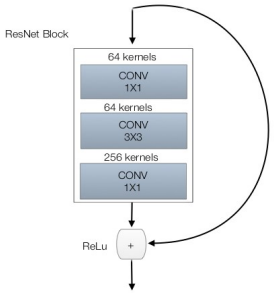
Figure 2.6: ResNet 50, 101 and 152 layered architectures[7]

The ResNet can be implemented with different number of layers, varying from 18 to 152 layers [6]. Each variant of ResNet does include a so-called ResBlock. These blocks either contain two layers (figure 2.7a) and only occur in the 18 and 34 ResNet variant or the blocks contain 3 layers (figure figure 2.7b) and only occur in the 50, 101 and 152 layer variant. The networks are able to skip over a block and immediately go to the ReLu operation. The opportunity to skip a block is only possible because the spatial size of the input of any block does not differ from the output of the block. The 18 and 34 layer ResNet variant are shown in figure 2.5. The difference between these two networks is not the architecture of the ResBlocks, but how often these blocks occur. In ResNet18 the blocks occur each two times. The first block of the ResNet34 variant occurs 3 times, the second ResBlock occurs 4 times, the third 6 times and the last 3 times. The 50, 101, and 152 layer variant are shown in figure 2.6. Again the difference between these networks is not in the architecture of the ResBlocks, but the amount of occurrences of the blocks. For ResNet50 the first block occurs 3 times, the second block 4 times, the third block 6 times and the last block 3 times. For ResNet101 only the third block differs from ResNet50 with 23 occurrences. In ResNet152 the second and third block occur more often than in the other networks, the second occurs 8 times and the third 36 times. Similar to other CNN's, ResNet architectures all include multiple convolutional layers. A convolution kernel has either the height and width of 7, 3 or 1 pixels. All ResNets also include a max pooling, an average pooling, a fully-connected, and a softmax layer.

[6]Image made by Anne-Ruth Meijer
[7]Image made by Anne-Ruth Meijer



(a) ResNet 2 layer block



(b) ResNet 3 layer block

Figure 2.7: The two different ResNet blocks[8]

2.2.8 DenseNet

A DenseNet utilizes the functionality of ResNets and also contain the possibility of skipping layers. However DenseNets contain parallel skips [7]. Because DenseNet contains some similar choices regarding the architecture, the vanishing gradient problem is also avoided. A DenseNet differs from a ResNet in the connections between the layers and the amount of layers. All layers in a DenseNet are all connected to every subsequent layer, meaning that every layer receives the activation maps of all preceding layers.
In figure 2.8 the architectures of the DenseNet121 and DenseNet161 are visualized, figure 2.9 shows DenseNet169 and DenseNet201. There are no differences in the architecture of the blocks, only in the amount of blocks. This is similar to the ResNet architectures. In these networks there are again multiple convolutional layers, an average pooling, a max pooling and a fully-connected layer. It also contains a ReLu operation after each convolutional layer. Because all features are send through all the following layers, the amount of input is much larger than the expected output. To reduce the number of input features 1x1 convolutions are used as a bottleneck before the 3x3 convolutions. In figure 2.8 and figure 2.9 there are also transition layers. These layers are also added to make sure that the network stays compact. Using multiple methods to ensure the model stays compact makes it possible that a network has a lot of layers.

[8]Image made by Anne-Ruth Meijer

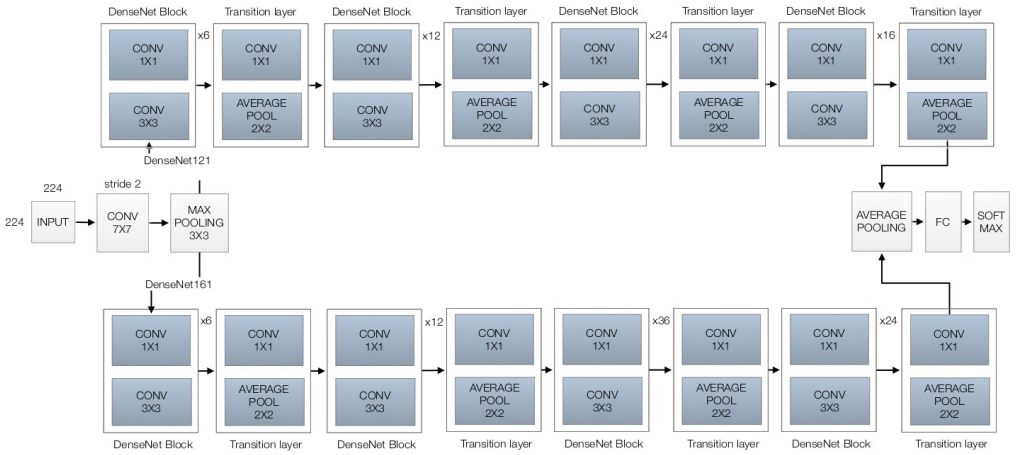


Figure 2.8: DenseNet architectures[9]

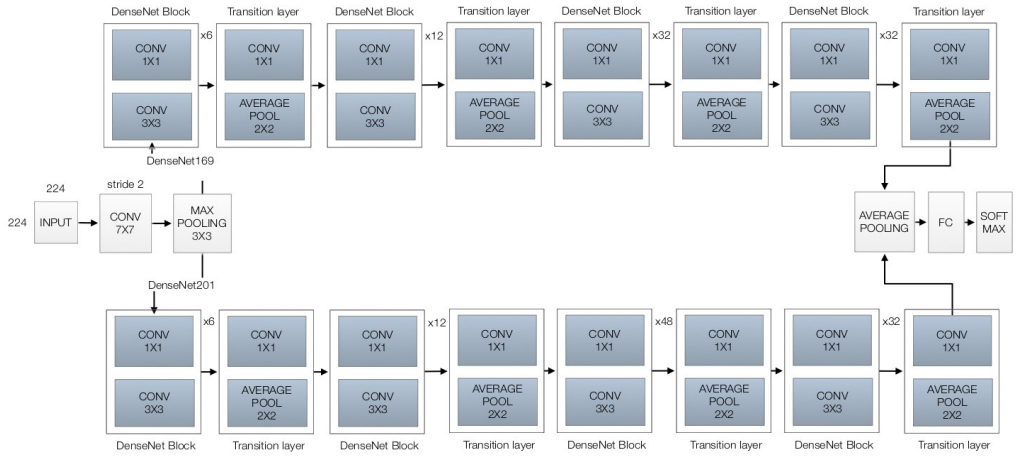


Figure 2.9: DenseNet architectures[10]

2.2.9 CORnet

The CORnet variations are developed to simulate the working of the human visual brain [12]. The variations all try to simulate four areas in the visual hierarchy: the V1, V2, V4 and IT region. There are three types of CORnets: the CORnet-S (skip), CORnet-Z (zero) and CORnet-R (recurrent). In this thesis only the CORnet-Z and CORnet-S variations are used. The networks are derived from the idea that AlexNet, a simple model, is nearly as good as deeper models like VGG. According to related work multiple fully connected layers do not seem necessary to get a good classification performance [14]. On these ideas the CORnet-Z is built with only a single convolution, followed by a ReLU and last a max pooling.
The recurrent version of the CORnets uses a biologically-valid unrolling to make sure it can be used for investigating neural dynamics. Biologic unrolling ensures a recurrent neural network can use information of features from a later stage in an earlier stage in the network [12]. Normally information from a later stage would be send to an earlier stage, but considering

[9]Image made by Anne-Ruth Meijer
[10]Image made by Anne-Ruth Meijer

the earlier stage is already finished with processing the input, the information would not be utilized. With biologically-valid unrolling networks the information is combined with the original input.
The skip version of the CORnet networks stacks two more convolutions on top of the recurrent version. This implementation draws inspiration of the ResNets. ResNets have shown to be one of the best models on behavioral benchmark [14]. The CORnet-S stacks two more convolutions, followed by normalizations on top of the CORnet-R structure. The second convolution expands the number of channels fourfold, which also happens in ResNet's 3 layered block structures. The last convolution decreases the number of channels back. The CorNet-S also includes a skip connection, similar to the skip connection in a ResNet or DenseNet.
The size of the input and kernels for the CORnet-Z and CORnet-S variant is shown in figure 2.10.

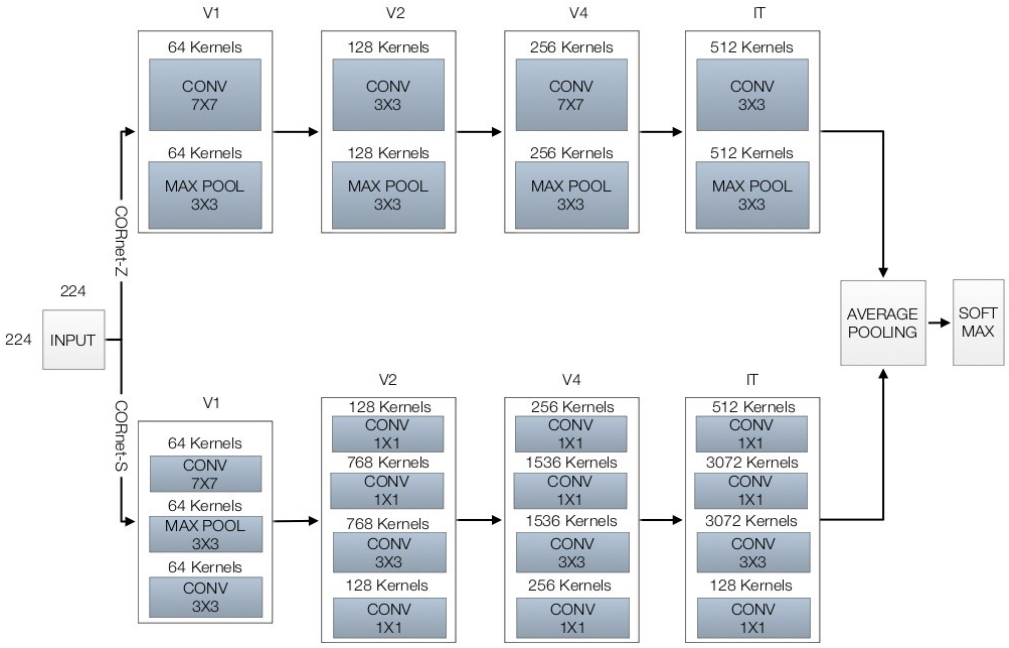


Figure 2.10: CORnet-S and CORnet-Z architectures[11]

2.3 Representational Similarity Analysis

To evaluate convolutional neural networks and the human visual brain, human fMRI scans and models constructed by networks have to be compared. This is possible by mapping the data from these scans and models to a common similarity space [10]. fMRI scans use the blood in the human brain to measure the activity of neurons in the brain [16], such a scan shows the reaction of the brain in response of viewing an image. In neural networks the reaction to an image is shown by the activations of the different layers. These activations show the reaction of the layers in response to the features of the image. To compare the effect of viewing an image in both a human and a network, the Representational Similarity Analysis (RSA) is an applicable method [10]. This method computes Representational Dissimilarity Matrices (RDMs) from the activation patterns of computational models and the brain. One

[11] Image made by Anne-Ruth Meijer

RDM represents the reaction of a network or brain to a set of images. The matrices display which images produce a similar reaction in the network or brain and which do not. For each image the dissimilarity between the activity patterns is computed by calculating one minus the Pearson correlation [10]. The Pearson correlation computes a correlation between 1 and -1 [1]. A value of 1 indicates that the two compared variables are perfectly positively linearly related. A value of -1 indicates that the compared variables are perfectly negatively linearly related. A value of 0 indicates no linear relation whatsoever. By computing the 1 minus Pearson correlation, a value of 0 or 2 will always indicate a perfect comparison and a value of 1 will therefore indicate no similarity at all. The formula of the Pearson correlation is shown in figure 2.11. Here n is the sample size, x_i and y_i are the individual samples, $\bar{x}$ and $\bar{y}$ stand for the mean of the sample.

$$r_{xy} = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sqrt{(\sum x_i^2 - n\bar{x}^2)}\sqrt{(\sum y_i^2 - n\bar{y}^2)}}.$$

Figure 2.11: Pearson correlation formula[12]

This method makes sure that both the mean level of activity and the variability of activity is measured. After computing the dissimilarity of all images, a matrix is constructed. This matrix is a NxN matrix, where N is the number of images, and the diagonal of the matrix only includes zeros. This is the case, because the matrix is a similarity matrix and the dissimilarity of the same activity patterns is always zero. The construction of RDMs is visualized in figure 2.12

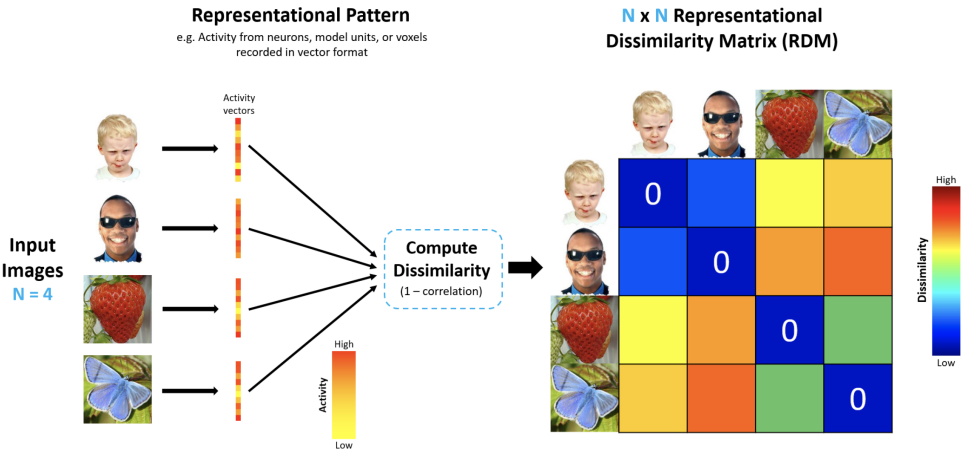


Figure 2.12: Generating RDMs[13]

[12] https://libguides.library.kent.edu/SPSS/PearsonCorr
[13] http://algonauts.csail.mit.edu/rsa.html

2.4 Related work

In order to simulate the human visual brain, the way humans recognize images need to be simulated. In recent years, deep convolutional neural networks have proven themselves to be quite good at recognizing patterns [13], these kind of networks have won numerous contests [8, 14]. Deep convolutional neural networks have also proven to be useful for simulation the human brain. Neural networks that have the same task as a brain, for example recognizing objects, are expected to contain mechanisms that are similar to the mechanisms in a brain. Earlier work of simulating the human visual brain is done by Brain-Score [14]. Brain-Score researched if artificial deep neural networks with improved task performance match the primate visual stream better than already existing networks. To answer the question they developed a method of scoring networks in brain similarity, called *Brain-Score* [14]. This scoring method is focused on the visual object recognition parts of the primate brain. The scoring method contains three brain benchmarks which eventually results in one score. Two of the benchmarks are neural benchmarks and one is behavioral. The neural benchmarks are constructed form neural recordings of primates, V4 and IT brain regions were used. The behavioral benchmark was obtained from human data. Against these benchmarks multiple networks were evaluated, to eventually find the best scoring and thus most "brain-like" network. After the initial research, Brain-Score still allowed third parties to submit new networks. As a result the best networks - described in the Brain-Score research paper [14] - are not currently the best. The known most brain-like networks according to the current Brain-Score ranking[14] are the DenseNet169, CorNet-S and the ResNet101. From this research not only brain-like networks were ranked, but also some interesting patterns were uncovered. For instance performing well on recognizing images does not necessarily correlate to performing well on brain simulation. This is proven by the fact that models scoring high on the ranks of ImageNet do not necessarily score high on the Brain-Score rank and vice versa. The results of the Brain-Score paper also show that high ImageNet performing networks score low on the behavioral benchmark. According to the paper this could be the reason that these networks score lower on the Brain-Score [14].

[14] http://www.brain-score.org/

Chapter 3

Approach

To conduct this research, different convolutional neural networks are tested. The networks include: VGG, AlexNet, ResNet18, ResNet34, ResNet50, ResNet101, ResNet152, DenseNet121, DenseNet161, DenseNet169, DenseNet201, CorNet-S, and CorNet-Z (see Appendix A-G). The networks are implemented using Pytorch. Pytorch has already trained models for VGG, AlexNet, the ResNet variations, and the DenseNet variations. The CorNet-S and CorNet-Z are also implemented in Pytorch. These network have also existing pre-trained models [15], although they aren't included in the standard pre-trained Pytorch models. After testing all the existing networks, the next step is training a new network based on the results. Based on the results a new ResNet variation is designed and trained, the results are described in Chapter 4. The Algonauts Project provided a development kit, training sets and test sets. The development kit includes sample code for generating activations from AlexNet, ResNet, and VGG, sample code for computing RDMs from these networks and code to evaluate these model RDMs.
This chapter gives an overview of what kind of data is used, the implementation details of all the networks and how the models are created, shows how the models are evaluated and finally explains the training process.

3.1 Data

The Algonauts Project provided three datasets: two training sets and one test set. The first training set includes 92 images along with human fMRI brain data RDMs from the EVC and IT region of the brain in response to viewing the 92 images. The second training set includes 118 images and again human fMRI brain data RDMs of the same regions in response to viewing the 118 images. These training sets are given to optimize the process of model selection. Meaning these sets can be used to test how well models can simulate the brain before choosing to submit a model. The third set is a test set and is used to submit models. For this reason there is no fMRI data publicly available for this set. The Algonauts Project has not provided any human RDMs, because they use this set to compare the submissions of all competitors in the challenge. In this research the training sets are used for comparing the networks and choosing a new architecture to train a new network. The test set is also used for testing the different networks, although these results will be used to test whether networks simulate the human visual brain.
The pre-trained CNN models all were trained on the ImageNet dataset, consisting of 1.2 milion images and 100.000 classes. To ensure that the pre-trained networks and the new trained network can be compared, the new trained network is trained on the same dataset.

[15] https://github.com/dicarlolab/CORnet

The dataset includes preprocessed images for training and validation.

3.2 Implementation

3.2.1 Networks

As mentioned the CNN's that are tested are all implemented using Pytorch with Python 3.7.3. Pytorch is an open source deep learning platform that has integrated Python, meaning that the libraries and packages of Python can be used for writing networks. To implement each network, first the standard structure of each network is defined in a separate file in a Python class. Here all the layers are constructed and the correct kernel size for each layer is defined. Normally each network would return a classification, but this research is interested in the activations of layers. These activations show the response of the network to viewing an image. In the forward section of the standard implementation of the networks every layer is called and the result of that layer is passed through the next layer. This is identical for every layer excluding the last, because the output of the last layer is a classification that is returned. In the implementation of the networks for the Algonauts challenge this is adjusted. In this case all the outputs of the different layers, excluding the last, is returned. The original implementation of the networks in Pytorch[16] is modified to return the activations. For the CORnet variations the situation is different, the implementation of these networks for the Algonauts challenge is constructed from the original implementation described in the original paper [12].
To test a pre-trained model with an image, first the model needs to be loaded in pytorch. When a model is loaded the input dataset is randomly shuffled and each image is re-sized and normalized. Every image returns an activation map for every layer in the network. These different activation maps are saved in a matlab structure per image.

3.2.2 Creating RDMs

When all activation maps are saved, they need to be processed into RDMs. Code for this is provided by the Algonauts challenge. However this is only usable with features saved in matlab structures. With the original implementation of the CORnet variations it is not possible to use this provided code. However with adding loops for some existing functions and changing the way the activations are saved this problem is resolved.
Since the activations of all layers can be computed, it is possible to assemble an RDM for every layer in every implemented network. In order to make a RDM for a layer, the features of that layer are collected and the Pearson correlation between those features is subtracted from 1. This is then added to the correct place in the RDM. The RDM that is constructed here is for both the EVC and IT region.
Although results of this research are the generated RDMs, they will not be shown. There are too many RDMs generated for all networks and they do not provide a more comprehensible image than the percentages calculated by the evaluation of the RDMs.

3.3 Evaluation

To test whether a network successfully simulates the human visual brain, generated RDMs will be compared with human data. The Algonauts Project evaluates the models by computing the noise normalized squared correlations of two models. The models being a RDM produced by a CNN and a RDM containing human fMRI data. The RDM containing human fMRI data is constructed from data of 15 humans. The reaction of 15 humans is recorded and a RDM is

[16] https://pytorch.org/docs/stable/torchvision/models.html

constructed for every participant. The average over the 15 RDMs is the target for a CNN to predict. There are two target RDMs for both the EVC and IT region for every dataset. The correlation that is used is the Spearman correlation [5]. The Spearman correlation is almost the same as the Pearson correlation. It is defined as the Pearson correlation, but between rank variables. Which means the strength of association between two variables is tested. This way one variable can be ranked better than the other. The Spearman correlation does not assume that the variables are linear, unlike the Pearson correlation. The correlation is a number between -1 and 1, these values reveal that there is a perfect correlation. The negative correlation implies that as the x-values increase the y-values decrease, while the positive correlation implies that as the x-values increase the y-values also increase. The value of 0 shows that there is no correlation at all.
The Spearman correlation is squared, ensuring there are no negative numbers. This value is then noise normalized, because the human brain data is limited by the noise during measurement and the amount of data. The Project understands the presence of noise and therefore they do not expect a correlation of 1. They calculated a noise ceiling for each provided dataset. The ceiling is the expected correlation that will be achieved by an ideal model given the data. The noise normalized values are represented in percentages.
The noise normalized squared Spearman correlation is computed for both the EVC and IT region of the brain. The average of these two scores is the score a model receives in the Algonauts challenge. The challenge not only provided a ceiling, but also a baseline. The baseline is the score a model is expected to beat to enter the competition, for this baseline the organizers utilized an AlexNet. The baseline precise values are mentioned in table 3.1. In this research the scoring technique of the challenge is used to score the CNN's, a model is assumed to simulate the human brain when the model scores better than the baseline.
To evaluate each model, first all the layers of one model are evaluated. When this is completed the layer scoring average the best on the EVC and IT region will be used to evaluate further. The best layers of the network are then compared on both the training sets and the test set.

EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
0.0042	6.58	0.0053	8.22	0.0048	7.41

Table 3.1: The baseline of the Algonauts challenge. R^2 is the squared Spearman correlation, nc is the noice normalized squared Spearman correlation

3.4 Self-trained network

3.4.1 Training

3.4.2 Architecture

ResNet20 is designed with the results of the pre-trained networks on the training datasets in mind. The ResNet has a similar architecture as ResNet18 and ResNet34, meaning the networks use the 2 layered ResBlocks. In ResNet20 the amount of occurences of the first ResBlock is the sameas the amount of occurences of the first ResBlock in ResNet34, it occurs 3 times. The rest of ResNet20 is identical to ResNet18, the architecture of ResNet20 is visualized in figure 3.1. The reason why only the first layer is changed is because of the performance of the first layer in ResNet18 and the performance of the first two layers in ResNet34. The first layer of ResNet18 is a lot better than the rest of the network, while in ResNet34 the performance of the first two layers is close. The expectation is that by only changing the first layer, the rest of the layers will improve the first layer. Thus expecting the same phenomenom as appeared in ResNet18. By adding an occurance of the first layer

(ResBlock) the layer gets a chance to train more. This could cause an even better simulating first layer.

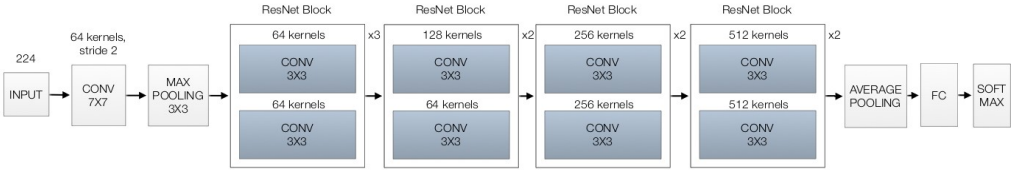


Figure 3.1: ResNet20 architecture [17]

[17] Image made by Anne-Ruth Meijer

Chapter 4

Results

This chapter evaluates the results of this research. First the results of the pre-trained networks on both the training and test sets are shown. Second the self-trained network is evaluated, the evaluated models include the trained model on epoch 1, 5, and 10. Last the performance on the EVC and IT regions by the networks are compared.

4.1 Pre-trained networks

Before the different pre-trained networks can be compared, the layers within each network need to be compared. The following results are all based on the tables in the appendices A to E. The best layer of a network is the layer that scored average the best on both training datasets. In table 4.1 the best layers of all pre-trained networks are shown. As explained in Chapter 2.2 the networks include different layers, the following layers are the layers tested during this research. The ResNet has 4 ResBlocks and a fully-connected layer, the VGG has 5 max pooling layers and 3 fully-connected layers, the AlexNet has 5 convolutional and 3 fully-connected layers, the DenseNets have 4 DenseBlocks, and the CORnets have a V1, V2, V4 and IT layer. Table 4.1 indicates that the best layers of the networks either are the first, second or fourth layer. The first layer is most often the best layer, with 5 occurrences. The second and fourth layer both are the best in four networks. The best layers of each network will from now on represent the network in this chapter.

Network	Best layer	Average nc over the training sets (%)
ResNet18	ResBlock 1	12.875
ResNet34	ResBlock 2	9.955
ResNet50	ResBlock 4	6.11
ResNet101	ResBlock 2	7.555
ResNet152	ResBlock 4	7.485
VGG	Max pooling 4	6.435
AlexNet	Convolution 2	8.9
DenseNet121	DenseBlock 1	7.725
DenseNet161	DenseBlock 1	9.035
DenseNet169	DenseBlock 1	10.065
DenseNet201	DenseBlock 1	6.8
CORnet-S	V2	3.095
CORnet-Z	V4	3.14

Table 4.1: The best layers average noise normalized squared correlation of the training sets of the pre-trained networks. nc is the noise normalized squared Spearman correlation

The average noise normalized squared Spearman correlation score over the training sets for all pre-trained networks is visualized in figure 4.1. The figure shows that the ResNet18 is the best average network on the training sets. The second best network is DenseNet169 and third the ResNet34. The figure also shows the standard deviation between the two sets. This shows that AlexNet is the most consistent over the two datasets and ResNet152 is the least consistent. The standard deviation is high for most networks, meaning that the results on the different datasets differ a lot. Figure 4.1 does not include a baseline, because there is no baseline for the training datasets.

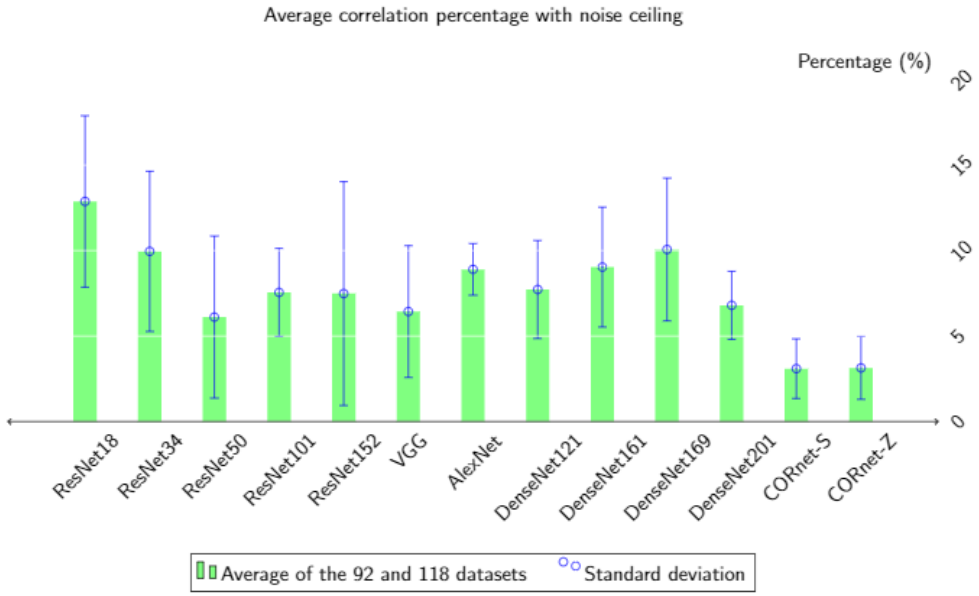


Figure 4.1: The average noise normalized squared Spearman correlation percentage of the Algonauts training sets for different CNN's, along with the standard deviation.

The best layers of the pre-trained networks are also submitted to the Algonauts challenge. In figure 4.2 the score of the submitted models at the Algonauts challenge is visualized, this figure is based on the results in appendix G. This figure includes the baseline for the challenge, because the baseline is based on the test set. The score is the average noise normalized squared Spearman correlation of the EVC and IT region for each network. The average highest scoring network is the DenseNet201, with second place ResNet18 and third ResNet34. Again the standard deviation is shown, this reveals that the DenseNet161 and CORnet-Z are the most consistent. ResNet18 also seems very consistent in the two brain regions, DenseNet201 reveals to be less consistent. The standard deviation still is small for almost all networks above the challenge, excluding DenseNet201. This means that the networks (one layer) almost always predict about the same for the EVC and IT region. ResNet18, ResNet34, ResNet101, DenseNet121, DenseNet161, and DenseNet201 score higher than the challenge baseline.

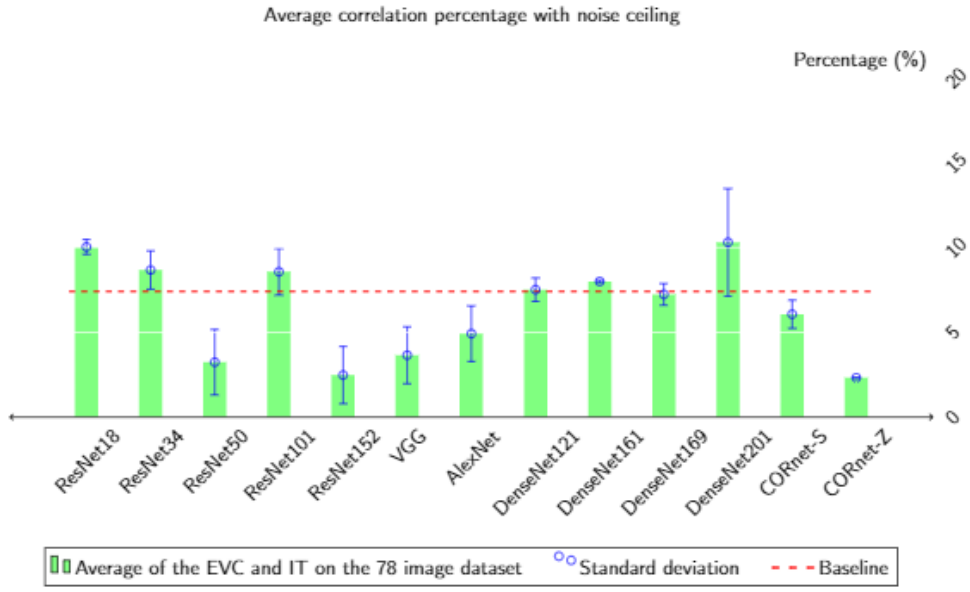


Figure 4.2: The noise normalized squared Spearman correlation percentage of the Algonauts test set for different CNN's, along with the standard deviation.

4.2 Self-trained network

After testing the different pre-trained networks a ResNet20 variant is trained. Three trained models are tested in this reasearch, a model trained on 1 epoch, on 5 epochs and on 10 epochs. The results of the three ResNet20 models are described in Appendix F. These tables show that the fully-connected layer is the average best layer for the three different models, these scores are generated from testing on the training sets. The models with the best layers and their scores are shown in table 4.2. This table shows that the ResNet20 trained with 10 epochs works best out of all ResNet20 models, although the variant trained on 1 epoch is a close second.

Network	Best layer	Score, average over the training sets (%)
ResNet20 epoch 0	fully-connected	10.325
ResNet20 epoch 5	fully-connected	5.945
ResNet20 epoch 10	fully-connected	10.645

Table 4.2: The best layers average noise normalized squared correlation of the training sets of the self-trained network models

The noise normalized squared correlation score and the standard deviation of the ResNet20 models is shown in figure 4.3. The ResNet20 trained for 10 epochs performs best on the training sets, yet it is also the model with the highest deviation. The network trained on 5 iterations has the lowest deviation. The standard deviation shows that the models trained on 5 and 10 epochs score a lot different on the different datasets.

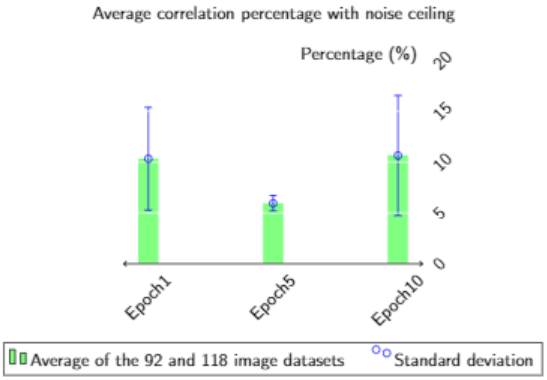


Figure 4.3: The average noise normalized squared Spearman correlation percentage of the Algonauts training sets for different ResNet20 models, along with the standard deviation

The models trained on different iterations were all submitted to the Algonauts challenge, the precise result are included in appendix G. Figure 4.4 shows these results, along with the standard deviation. Similar to the results on the training sets, the model trained for 10 epochs performs the best. All the models produce results above the baseline. Also, the standard deviation is less in the results of the test set than the result of the training sets. This is consistent with the standard deviations of the pre-trained models. It again shows that a layer of a network achieves almost similar scores on the different brain regions. The model that is the most consistent is again the model trained for 5 epochs.

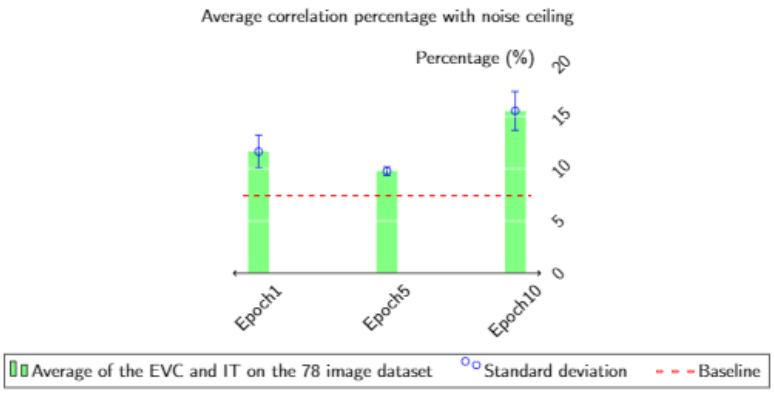


Figure 4.4: The average noise normalized squared Spearman correlation percentage of the EVC and IT of the test set for different ResNet20 models, along with the standard deviation

4.3 Networks above baseline

In figure 4.2 the average score of networks above the baseline can be perceived. From this figure the ResNet18, ResNet34, ResNet101, DenseNet121, DenseNet161, and DenseNet201 seem to be the networks that score above the baseline. However as mentioned this is the average score, the networks could score above the baseline of a brain region while not scoring high enough on average. The networks scoring above the baseline on the EVC region is shown in figure 4.5, for the IT region this is visualized in figure 4.6. These figures are based on the results in appendix G. The networks that score above the baseline for the IT region also score above baseline on average. However the networks scoring above baseline in the EVC region also include the DenseNet121 and DenseNet161. The best scoring network for both regions is ResNet20 trained for 10 epochs.

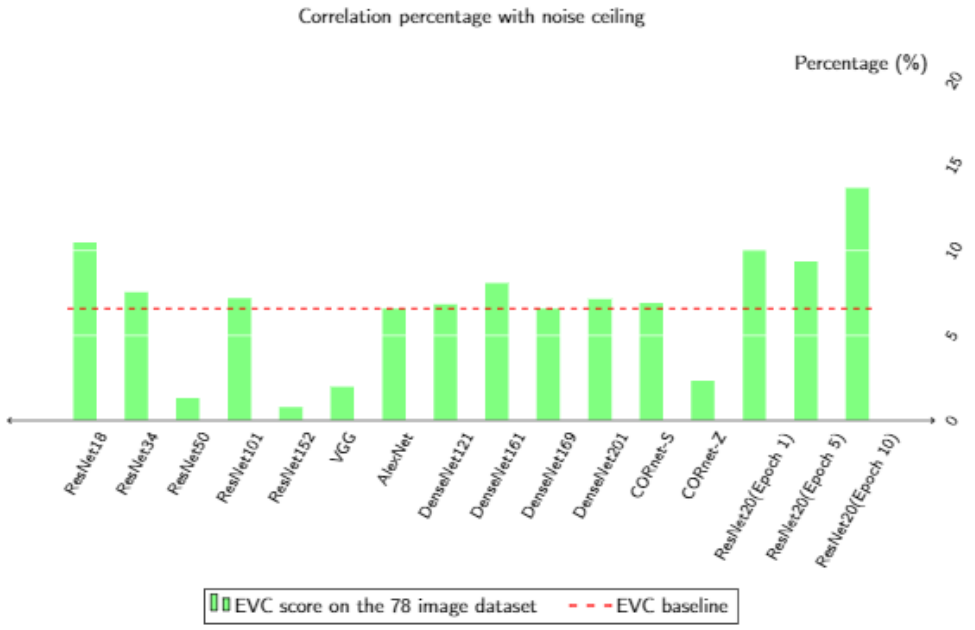


Figure 4.5: The noise normalized squared Spearman correlation percentage for the EVC brain region on the Algonauts test set for the networks above the challenge baseline

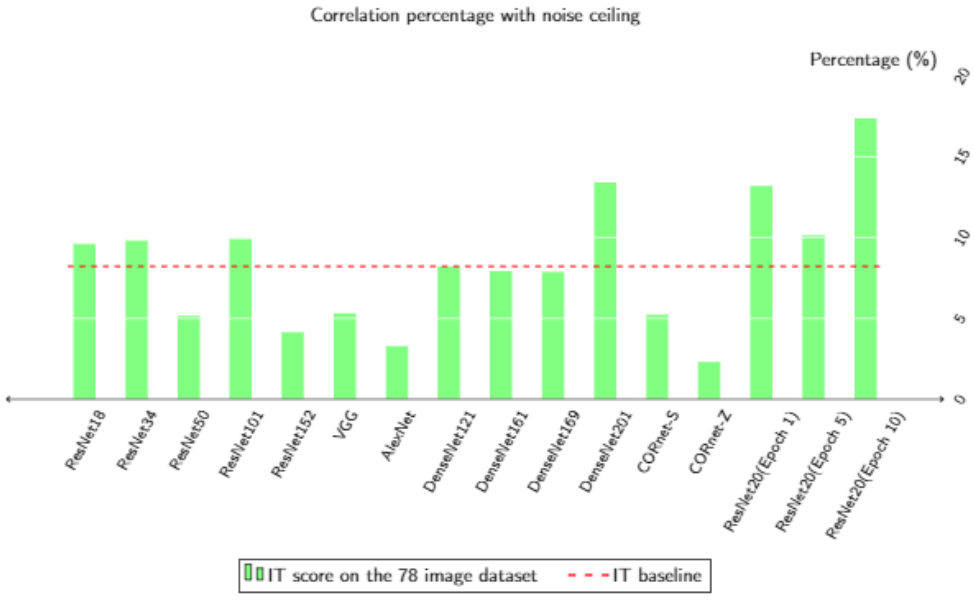


Figure 4.6: The noise normalized squared Spearman correlation percentage for the IT brain region on the Algonauts test set for the networks above the challenge baseline

Chapter 5

Discussion

The results in chapter 4 revealed some interesting properties of the CNN's, which are discussed in this chapter. The results show that ResNet20 trained on 10 epochs is the best performing network. This is the case in both the EVC and IT region of the brain, an unexpected result. The different brain regions are responsible for different parts of the visual hierarchy, so it is unexpected that one layer performs best on different regions. This is not only observed in ResNet20 trained on 10 epochs, but also in other networks. The standard deviations between the different brain regions on the test data indicate that a lot of networks obtain about the same score for each region in the brain. The majority of CNN's perform best in their first or second layer according to the results on the training data. It is interesting that both regions are often best predicted by the first layers. As mentioned in Chapter 2.1 the EVC is responsible for combining information of the left and right eye and also plays a part in processing properties of a viewed object. The IT region is also part of this process, but appears later in the visual hierarchy. In a CNN earlier layers are responsible for early recognition of obvious patterns, this corresponds to the functions of the EVC region. Later layers in CNN's are trained for recognizing more specific patterns in images, corresponding to the IT region. The IT region occurs later in the hierarchy, so the expectation would be that this region would be better simulated by later layers. However both the EVC and the IT regions contain parts that are responsible for processing properties of images, so it is not abnormal that both regions are well predicted by the first layers.

The best performing model ResNet20 is designed with the results of the other networks in mind. These results showed that the first layer of ResNet18 and second layer of ResNet34 were the superior layers of their networks. By combining their architectures the fully-connected layer thrived. This could be caused by the different architecture or the difference in training time, since the training time of the self-trained network is only 10 epochs. In the pre-trained networks the best scoring layers all are ResBlocks, while the self-trained network achieves the highest scores with the fully-connected layer. Cause for not well working early layers could be the lack of training time, since earlier layers get better in recognizing patterns with more training time. An explanation for the performance of the fully-connected layer could also be the lack of a lot of epochs. In the fully-connected layers all important features are combined. With only a few epochs these features would be the obvious ones. If these features are enough for a human to recognize an object, maybe the brain only reacts to them. This could cause a similar reaction in the layer and the brain. The conclusion that the fully-connected layer works better with less training time can be confirmed by self-training the other networks and testing the layers with the same amount of epochs.

An explanation why ResNet20 outperforms other longer trained networks could be that

performance on recognizing images on the ImageNet data does not directly correlate with performance on simulating the human visual brain. This is what Brain-Score also discussed in their research [14].

The results on the two different training sets show that the standard deviation between the sets is large, meaning that some networks perform very different on various data. Cause for this could be that the two datasets are so small they differ too much from each other. Both datasets only contain around 100 images and thus can cause results that will not hold in real life. Since the test set is smaller than 100 images, the networks are tested on a delimited world. To make sure the best simulating networks really simulate the human brain a larger more lifelike dataset of images should be used for testing.

Even though there are some uncertainties, the result are promising. A shallow network (scoring 15.53%) proves to be outperforming deeper networks like DenseNet201 (scoring 10.31%) in similar situations. As explained, without further training and testing it is not possible to conclude if this improvement is due to the architecture or training time. However the results demonstrate that the DenseNet201 is already outperformed after one epoch, this questions the impact of the training time and thus training time should not be overestimated.

Chapter 6

Conclusion

The aim of this research is trying to understand and thus simulate the human visual brain. Therefore the research question is: "Can the brain activity in the EVC and IT regions of the brain be predicted in response to viewing an image?". This question results in multiple sub-questions: "Are deep CNN's better for simulating the human visual brain than shallow CNN's?", "Does training improve the ability of a CNN to simulate the human visual brain?", "Can the brain activity in the EVC region of the brain be predicted in response to viewing an image?", "Can the brain activity in the IT region of the brain be predicted in response to viewing an image?", "Which architecture is best for simulating the human visual brain?". In order to answer these questions the conclusion of when a CNN simulates the brain needs to be measurable. The measurement tool in this research is the noise normalized squared Spearman correlation score of the generated model and fMRI data. When the score is higher than the Algonauts challenge baseline, a model is considered to be simulating the human brain.

The best performing network in the challenge is ResNet20, the second best is DenseNet201 and third comes ResNet18. The best performing network does have significantly less layers than the second best. This is also observed in the evaluation of the networks on the training sets, where ResNet18 performs best. These results indicate that shallow networks outperform deeper neural networks when the task is simulating the human visual brain.
As mentioned is the ResNet20 tested in three different variations, with 1, 5 and 10 epochs. The results indicate that the short-trained models of this network are not outperformed by the longer trained pre-trained networks. Moreover the self-trained network trained with 10 epochs is as it happens the best performing network, scoring highest for both the EVC and IT region. All the pre-trained networks are trained with more epochs. Revealing that a longer training time does not necessarily equates to a better brain simulating network.
There are multiple networks that score above the baseline for the EVC region and for this reason simulate the EVC region of the brain. The IT region is also simulated by multiple networks, although less than the EVC region. This means that the EVC and IT region can be separately predicted by multiple CNN's. The networks best simulating the EVC region is ResNet20, this networks also best simulates the IT region. For the EVC region ResNet18 performs second best, while DenseNet201 performs second best on the IT region. The overall best scoring network is also the ResNet20 trained on 10 epochs. Meaning that the ResNet20 architecture is the best architecture for simulating the human visual brain.

The answer to the question: "Can the brain activity in the EVC and IT regions of the brain be predicted in response to viewing an image?" is yes. Multiple networks have scored better than the Algonauts challenge baseline and thus simulate the human visual brain.

The networks that simulate the human visual brain are ResNet18, ResNet34, ResNet101, DenseNet121, DenseNet161, DenseNet201 and the three trained variations of ResNet20.

ResNet20 is the best performing network tested by this research with an average score of 15.53%. This shallow network outperforms deeper networks with over 200 layers, while only being trained for 10 epochs. Since the baseline is 7.41%, this network is considered simulating the human brain according to the Algonauts challenge.

6.1 Future research

In this research multiple networks are compared to each other and from that comparison a new network is trained. Training that network further and experimenting more with its architecture would be interesting. This could show if the network would become better with further training and how many epochs would be optimal. Also experimenting with the architecture could indicate what about the architecture makes it a well simulating network. Another interesting research would be if the networks best simulating the regions in the ventral stream also work well simulating the dorsal stream. Being responsible for the spatial information of viewed objects and not the properties, it is possible that other networks work better for that stream. The spatial information of an object is different from the properties like color and shape, networks more focused on recognizing spatial aspects of an image could be better equipped for simulating the dorsal stream.
This research is based on the Algonauts challenge 2019, which is part of the Algonauts Project. The Algonauts Project will launch more challenges in their quest to understand human intelligence. An interesting challenge after this would be to recognize actions. This would be a challenge trying to simulate the human visual brain not in response to viewing a static image, but in response to viewing moving images such as video's.

Bibliography

[1] J. Benesty, J. Chen, Y. Huang and I. Cohen, "Pearson correlation coefficient", in "Noise reduction in speech processing", pp. 1–4, Springer, 2009.

[2] J. Blumberg and G. Kreiman, "How cortical neurons help us see: visual recognition in the human brain", *The Journal of clinical investigation*, volume 120(9):pp. 3054–3063, 2010.

[3] I. Charest, R. A. Kievit, T. W. Schmitz, D. Deca and N. Kriegeskorte, "Unique semantic space in the brain of each beholder predicts perceived similarity", *Proceedings of the National Academy of Sciences*, volume 111(40):pp. 14565–14570, 2014.

[4] R. M. Cichy, G. Roig, A. Andonian, K. Dwivedi, B. Lahner, A. Lascelles, Y. Mohsenzadeh, K. Ramakrishnan and A. Oliva, "The Algonauts Project: A Platform for Communication between the Sciences of Biological and Artificial Intelligence", *arXiv e-prints*, arXiv:1905.05675, May 2019.

[5] J. Hauke and T. Kossowski, "Comparison of values of Pearson's and Spearman's correlation coefficients on the same sets of data", *Quaestiones geographicae*, volume 30(2):pp. 87–93, 2011.

[6] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", *CoRR*, volume abs/1512.03385, 2015.

[7] G. Huang, Z. Liu and K. Q. Weinberger, "Densely Connected Convolutional Networks", *CoRR*, volume abs/1608.06993, 2016.

[8] K. M. Jozwik, N. Kriegeskorte, K. R. Storrs and M. Mur, "Deep convolutional neural networks outperform feature-based but not categorical models in explaining object similarity judgments", *Frontiers in psychology*, volume 8:p. 1726, 2017.

[9] J. N. Kok, E. J. Boers, W. A. Kosters, P. Van der Putten and M. Poel, "Artificial intelligence: definition, trends, techniques, and cases", *Artificial intelligence*, volume 1, 2009.

[10] N. Kriegeskorte, M. Mur and P. A. Bandettini, "Representational similarity analysis-connecting the branches of systems neuroscience", *Frontiers in systems neuroscience*, volume 2:p. 4, 2008.

[11] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", in "Advances in Neural Information Processing Systems 25", (edited by F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger), pp. 1097–1105, Curran Associates, Inc., 2012.

[12] J. Kubilius, M. Schrimpf, A. Nayebi, D. Bear, D. L. K. Yamins and J. J. DiCarlo, "CORnet: Modeling the Neural Mechanisms of Core Object Recognition", *bioRxiv*, 2018, doi:10.1101/408385.

[13] J. Schmidhuber, "Deep learning in neural networks: An overview", *Neural Networks*, volume 61:pp. 85 – 117, 2015, ISSN 0893-6080, doi:https://doi.org/10.1016/j.neunet.2014.09.003.

[14] M. Schrimpf, J. Kubilius, H. Hong, N. J. Majaj, R. Rajalingham, E. B. Issa, K. Kar, P. Bashivan, J. Prescott-Roy, K. Schmidt, D. L. K. Yamins and J. J. DiCarlo, "Brain-Score: Which Artificial Neural Network for Object Recognition is most Brain-Like?", *bioRxiv preprint*, 2018.

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

[16] K. Smith, "Brain imaging: FMRI 2.0", *Nature News*, volume 484(7392):p. 24, 2012.

[17] Stanford-University, "Cs231n: Convolutional Neural Networks for Visual Recognition", 2019, [Online; accessed 11-juni-2019].

Appendix A

ResNet tables

The evaluation of the ResNet18 (nc stands for noise ceiling):

block 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0285	17.94	0.0082	2.66	0.0183	7.86
118 Images	0.0257	24.50	0.0061	8.39	0.0159	17.89
block 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0130	8.18	0.0037	1.21	0.0084	3.59
118 Images	0.0138	13.13	0.0031	4.29	0.0084	9.50
block 3						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0083	5.28	0.0063	2.05	0.0073	3.15
118 Images	0.0026	2.52	0.0016	2.22	0.0021	2.39
block 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0143	8.98	0.0387	12.57	0.0265	11.35
118 Images	0.0006	0.56	0.0011	1.53	0.0009	0.96
fc						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0042	2.67	0.0069	1.91	0.0051	2.17
118 Images	0.0024	2.27	0.0019	2.66	0.0022	2.43

The evaluation of the Resnet34 (nc stands for noise ceiling):

block 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0229	14.44	0.0052	1.68	0.0141	6.03
118 Images	0.0158	15.09	0.0057	7.86	0.0108	12.13
block 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0191	12.06	0.0054	1.76	0.0122	5.27
118 Images	0.0221	21.11	0.0039	5.32	0.0130	14.64
block 3						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0130	8.17	0.0051	1.67	0.0091	3.88
118 Images	0.0044	4.19	0.0027	3.71	0.0035	3.99
block 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0178	11.21	0.0319	10.36	0.0248	10.65
118 Images	0.0006	0.54	0.0012	1.69	0.0009	1.01
fc						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0045	2.85	0.0061	1.97	0.0053	2.27
118 Images	0.0032	3.05	0.0022	3.01	0.0027	3.04

The evaluation of the Resnet50 (nc stands for noise ceiling):

block 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0136	8.59	0.0047	1.53	0.0092	3.94
118 Images	0.0027	2.59	0.0032	4.37	0.0029	3.32
block 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0172	10.80	0.0039	1.28	0.0106	4.53
118 Images	0.0104	9.93	0.0032	4.38	0.0068	7.66
block 3						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0118	7.45	0.0048	1.55	0.0083	3.56
118 Images	0.0021	1.98	0.0025	3.49	0.0023	2.60
block 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0198	12.45	0.0308	10.02	0.0253	10.85
118 Images	0.0013	1.28	0.0011	1.50	0.0012	1.37
fc						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0053	3.31	0.0056	1.84	0.0055	2.34
118 Images	0.0025	2.43	0.0020	2.68	0.0023	2.53

The evaluation of the Resnet101 (nc stands for noise ceiling):

block 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0199	12.54	0.0099	3.23	0.0149	6.40
118 Images	0.0017	1.66	0.0032	4.45	0.0025	2.80
block 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0196	12.32	0.0036	1.18	0.0116	4.98
118 Images	0.0128	12.19	0.0052	7.16	0.0090	10.13
block 3						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0229	14.44	0.0083	2.69	0.0156	6.70
118 Images	0.0049	4.69	0.0044	6.11	0.0047	5.27
block 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0237	14.93	0.0252	8.18	0.0244	10.48
118 Images	0.0006	0.62	0.0013	1.73	0.0010	1.08
fc						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0064	4.03	0.0058	1.89	0.0061	2.62
118 Images	0.0021	1.97	0.0017	2.36	0.0019	2.12

The evaluation of the Resnet152 (nc stands for noise ceiling):

block 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0131	8.27	0.0062	2.01	0.0097	4.14
118 Images	0.0030	2.86	0.0034	4.68	0.0032	3.60
block 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0160	10.09	0.0032	1.04	0.0096	4.12
118 Images	0.0134	12.77	0.0049	6.73	0.0091	10.30
block 3						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0151	9.51	0.0128	4.17	0.0140	5.99
118 Images	0.0069	6.62	0.0042	5.79	0.0056	6.28
block 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0248	15.63	0.0406	13.20	0.0327	14.03
118 Images	0.0006	0.56	0.0011	1.47	0.0008	0.94
fc						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0061	3.86	0.0072	2.33	0.0066	2.85
118 Images	0.0028	2.68	0.0018	2.50	0.0023	2.60

Appendix B

VGG tables

The evaluation of the VGG (nc stands for noise ceiling):

maxpool 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0115	7.24	0.0043	1.40	0.0079	3.39
118 Images	0.0036	3.44	0.0006	0.89	0.0021	2.39
maxpool 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0159	10.03	0.0059	1.92	0.0109	4.68
118 Images	0.0013	1.27	0.0009	1.30	0.0011	1.29
maxpool 3						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0228	14.33	0.0146	4.76	0.0187	8.02
118 Images	0.0019	1.80	0.0010	1.32	0.0014	1.60
maxpool 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0250	15.74	0.0230	7.48	0.0240	10.29
118 Images	0.0036	3.46	0.0009	1.30	0.0023	2.58
maxpool 5						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0151	9.50	0.0155	5.03	0.0153	6.56
118 Images	0.0013	1.27	0.0019	2.59	0.0016	1.81
f6						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0066	4.14	0.0060	1.96	0.0063	2.70
118 Images	0.0002	0.19	0.0008	1.12	0.0005	0.57
f7						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0046	2.89	0.0109	3.54	0.0077	3.32
118 Images	0.0002	0.18	0.0010	1.40	0.0006	0.68
f8						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0037	2.34	0.0072	2.33	0.0054	2.33
118 Images	0.0026	2.44	0.0021	2.90	0.0023	2.63

Appendix C

AlexNet tables

The evaluation of the Alexnet (nc stands for noise ceiling):

Conv 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0109	6.84	0.0039	1.25	0.0074	3.16
118 Images	0.0016	1.50	0.0009	1.24	0.0012	1.39
Conv 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0267	16.79	0.0219	7.11	0.0243	10.41
118 Images	0.0117	11.20	0.0014	1.91	0.0066	7.39
Conv 3						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0284	17.85	0.0219	7.14	0.02516	10.79
118 Images	0.0075	7.13	0.0006	0.82	0.0040	4.55
Conv 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0222	13.95	0.0328	10.66	0.0275	11.79
118 Images	0.0024	2.28	0.0008	1.10	0.0016	1.79
Conv 5						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0128	8.03	0.0134	4.34	0.0131	5.60
118 Images	0.0012	1.14	0.0009	1.27	0.0011	1.19
f6						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0052	3.24	0.0265	8.62	0.0158	6.79
118 Images	0.0007	0.65	0.0012	1.70	0.0010	1.08
f7						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0056	3.53	0.0263	8.56	0.0160	6.84
118 Images	0.0007	0.68	0.0013	1.83	0.0010	1.15
f8						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0039	2.47	0.0111	3.61	0.0075	3.22
118 Images	0.0024	2.30	0.0031	4.22	0.0027	3.09

Appendix D

DenseNet tables

The evaluation of the DenseNet121 (nc stands for noise ceiling):

Denseblock 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0184	11.61	0.0042	1.37	0.0113	4.86
118 Images	0.0132	12.56	0.0056	7.75	0.0094	10.59
Denseblock 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0107	6.76	0.0050	1.62	0.0079	3.37
118 Images	0.0075	7.11	0.0021	2.93	0.0048	5.40
Denseblock 3						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0153	9.63	0.0065	2.12	0.0109	4.68
118 Images	0.0013	1.28	0.0015	2.02	0.0014	1.58
Denseblock 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0162	10.18	0.0267	8.69	0.0214	9.20
118 Images	0.0025	2.35	0.0017	2.32	0.0021	2.34

The evaluation of the DenseNet161 (nc stands for noise ceiling):

Denseblock 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0209	13.41	0.0049	1.60	0.0129	5.53
118 Images	0.0175	16.69	0.0048	6.57	0.0111	12.54
Denseblock 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0091	5.75	0.0079	2.57	0.0085	3.65
118 Images	0.0020	1.91	0.0023	3.22	0.0022	2.45
Denseblock 3						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0137	8.60	0.0064	2.09	0.0100	4.31
118 Images	0.0015	1.48	0.0023	3.21	0.0019	2.19
Denseblock 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0090	5.64	0.0126	4.08	0.0108	4.61
118 Images	0.0015	1.47	0.0017	2.28	0.0016	1.80

The evaluation of the DenseNet169 (nc stands for noise ceiling):

Denseblock 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0221	13.92	0.0054	1.74	0.0137	5.89
118 Images	0.0197	18.76	0.0056	7.74	0.0126	14.24
Denseblock 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0103	6.50	0.0066	2.15	0.0085	3.63
118 Images	0.0028	2.65	0.0017	2.34	0.0022	2.52
Denseblock 3						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0147	9.24	0.0069	2.24	0.0108	4.63
118 Images	0.0013	1.22	0.0022	2.99	0.0017	1.94
Denseblock 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0083	5.25	0.0155	5.03	0.0119	5.11
118 Images	0.0016	1.50	0.0018	2.46	0.0017	1.89

The evaluation of the DenseNet201 (nc stands for noise ceiling):

Denseblock 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0183	11.49	0.0047	1.36	0.0112	4.81
118 Images	0.0102	9.73	0.0054	7.44	0.0078	8.79
Denseblock 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0103	6.45	0.0055	1.79	0.0079	3.38
118 Images	0.0068	6.48	0.0022	3.05	0.0045	5.08
Denseblock 3						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0159	10.03	0.0097	3.15	0.0128	5.50
118 Images	0.0010	0.96	0.0011	1.48	0.0010	1.17
Denseblock 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0118	7.43	0.0184	6.00	0.0151	6.48
118 Images	0.0016	1.56	0.0017	2.38	0.0017	1.89

Appendix E

CORnet tables

The evaluation of the CORnet-Z (nc stands for noise ceiling):

V1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0014	0.88	0.0025	0.82	0.0020	0.84
118 Images	0.0004	0.42	0.0008	1.07	0.0006	0.69
V2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0028	1.76	0.0033	1.06	0.0030	1.30
118 Images	0.0044	4.21	0.0028	3.83	0.0036	4.06
V4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0027	1.72	0.0033	1.09	0.0030	1.30
118 Images	0.0045	4.32	0.0043	5.92	0.0044	4.98
IT						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0020	1.28	0.0038	1.25	0.0029	1.26
118 Images	0.0015	1.45	0.0018	2.50	0.0017	1.88

The evaluation of the CORnet-S (nc stands for noise ceiling):

V1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0019	1.17	0.0024	0.79	0.0021	0.92
118 Images	0.0033	3.18	0.0020	2.68	0.0026	2.98
V2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0026	1.66	0.0037	1.19	0.0032	1.35
118 Images	0.0045	4.29	0.0041	5.63	0.0043	4.84
V4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0016	0.99	0.0031	1.02	0.0024	1.01
118 Images	0.0028	2.69	0.0021	2.90	0.0025	2.78
IT						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0016	1.01	0.0068	2.23	0.0042	1.81
118 Images	0.0009	0.82	0.0010	1.31	0.0025	2.78

Appendix F

ResNet20 tables

The evaluation of the ResNet20 1 epoch (nc stands for noise ceiling):

block 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0092	5.76	0.0071	2.30	0.0081	3.48
118 Images	0.0019	1.81	0.0017	2.28	0.0018	2.00
block 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0139	8.74	0.0099	3.23	0.0119	5.11
118 Images	0.0033	3.13	0.0028	3.88	0.0030	3.43
block 3						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0165	10.36	0.0119	3.86	0.0142	6.08
118 Images	0.0037	3.53	0.0033	4.55	0.0035	3.95
block 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0038	2.42	0.0034	1.10	ß0.0036	1.55
118 Images	nan	nan	nan	nan	nan	nan
fc						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0352	22.17	0.0365	11.87	0.0359	15.38
118 Images	0.0053	5.07	0.0040	5.55	0.0047	5.27

The evaluation of the ResNet20 5 epochs (nc stands for noise ceiling):

block 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0092	5.81	0.0071	2.32	0.0082	3.51
118 Images	0.0019	1.86	0.0017	2.35	0.0018	2.06
block 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0140	8.81	0.0099	3.23	0.0120	5.13
118 Images	0.0034	3.22	0.0029	3.97	0.0031	3.53
block 3						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0167	10.50	0.0119	3.87	0.0143	6.13
118 Images	0.0038	3.62	0.0034	4.61	0.0036	4.02
block 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0040	2.53	0.0058	1.89	ß0.0049	2.11
118 Images	nan	nan	nan	nan	nan	nan
fc						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0169	10.62	0.0144	4.69	0.0156	6.71
118 Images	0.0054	5.14	0.0038	5.24	0.0046	5.18

The evaluation of the ResNet20 10 epochs (nc stands for noise ceiling):

block 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0093	5.87	0.0072	2.34	0.0083	3.54
118 Images	0.0021	1.96	0.0019	2.61	0.0020	2.23
block 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0136	8.58	0.0098	3.20	0.0171	5.03
118 Images	0.0031	2.99	0.0031	4.22	0.0031	3.49
block 3						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0153	9.64	0.0113	3.69	0.0133	5.71
118 Images	0.0034	3.25	0.0035	4.84	0.0035	3.90
block 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0121	7.63	0.0165	5.36	0.0143	6.13
118 Images	nan	nan	nan	nan	nan	nan
fc						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
92 Images	0.0402	25.28	0.0370	12.04	0.0386	16.55
118 Images	0.0045	4.27	0.0039	5.42	0.0042	4.74

Appendix G

Algonauts challenge result table

ResNet18 block 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0067	10.48	0.0062	9.60	0.0065	10.04
ResNet34 block 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0048	7.56	0.0063	9.80	0.0056	8.69
ResNet50 block 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0008	1.31	0.0033	5.15	0.0021	3.24
ResNet101 block 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0046	7.20	0.0064	9.92	0.0055	8.57
ResNet152 block 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0005	0.79	0.0027	4.15	0.0016	2.48
VGG maxpool 4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0013	1.97	0.0034	5.32	0.0024	3.65
AlexNet conv 2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0042	6.58	0.0021	3.28	0.0032	4.92
DenseNet121 denseblock 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0044	6.83	0.0053	8.21	0.0048	7.52
DenseNet161 denseblock 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0052	8.08	0.0051	7.92	0.0051	8.00
DenseNet169 denseblock 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0042	6.58	0.0051	7.87	0.0047	7.24
DenseNet201 denseblock 1						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0046	7.14	0.0087	13.43	0.0066	10.31
CORnet-S V2						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0044	6.90	0.0034	5.24	0.0039	6.07
CORnet-Z V4						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0015	2.33	0.0015	2.32	0.0015	2.33
ResNet20 epoch 1 fc						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0064	10.07	0.0085	13.20	0.0075	11.64
ResNet20 epoch 5 fc						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0060	9.36	0.0066	10.15	0.0063	9.76
ResNet20 epoch 10 fc						
Data	EVC R^2	EVC nc (%)	IT R^2	IT nc (%)	Average R^2	Average nc (%)
78 Images	0.0088	13.67	0.0112	17.37	0.0100	15.53