# Implementing Artificial Neural Networks with Tensorflow

Final Project Report

Büürma, Anneke 995025

Amme, Carmen 994813

WS 2022/2023

## 1 Introduction and Motivation

- *The context and Motivation for your project (What is your project? What ideas/problems is it based on? Why is the problem you are working relevant?)*
- *The related literature to your project: What other existing literature and projects is your project based on? What problem are you solving, what is known about this kind of problem, how have publications already tackled it (or similar problems)? Notice this is a major point for grading.*

Modeling the primate visual system and perception and thereby gaining understanding about its functioning has been an ongoing effort spanning different disciplines, including neuroscience and computer science (Lindsay, 2021). With recent increases in computational power, deep neural networks (DNNs) have been successfully used to model the human visual system, thereby revealing new insights about its internal functioning (e.g., Kietzmann, McClure, et al., 2019; Yamins et al., 2014).

Early models of the neural system used simple calculations within shallow network architectures, leaving them unable to perform complex tasks such as visual object recognition, a task that the human visual system does with ease (Kriegeskorte, 2015). Because the human visual system can be seen as a dynamic and complex deep recurrent neural network (Kietzmann, McClure, et al., 2019), early models of the visual system were not complex enough to account for the multiple processing stages needed for visual perception. Since the introduction of AlexNet, a deep convolutional neural network (CNN) trained on image classification, significant progress has been made in building computational models that perform on or even surpass human level, which was made possible through the increasing availability of data, a larger network size with shared weights, and a reduced training time (He et al., 2015a; Krizhevsky et al.,

2012). The visual system transforms and processes incoming information efficiently for organisms to behave adaptively (Yamins and DiCarlo, 2016). This transformation process can be untangled and modeled through CNNs.

Higher visual areas, like V4 and IT, located in the ventral stream and involved in object recognition, are complicated to understand (Yamins et al., 2014). When aiming to disentangle such neural activity, the larger number of parameters that follows from an increased complexity of modern CNNs is necessary to capture the vast amount of knowledge that is necessary for modeling intelligent behavior and visual processes (Kietzmann, McClure, et al., 2019). This development led to an accurate prediction of V4 and IT neuronal activity in humans (Yamins et al., 2014). The optimization of task performance to match human behavior shaped the connectivity patterns within the network, a constraint that might have also shaped the human visual cortex. This example showed how CNNs, whose standard architectures have been inspired by basic functional mechanisms of the primate visual system discovered by neuroscientists (Lindsay, 2021), accurately predicted complex neural data in higher visual areas and shaped novel ideas about the functioning of the visual system. Further improvement of models of the primate visual system that will be better at predicting human brain data and behavior will increase our understanding of the processing of how humans perceive the world.

It is known that the primate visual stream represents information hierarchically with an increasing level of complexity to solve the objective of object representation (**doi:10.1126/science.1117593** ; Hubel and Wiesel, 1962; Tanaka, 1996). More recent studies have found that DNNs show an increase in complexity with increasing processing that matches that of the ventral visual stream in the primate cortex (Güçlü and van Gerven, 2015), which consists of a number of areas involved in object recognition (Goodale and Milner, 1992).

Within this final project, we compare representations of visual content at different stages between three different DNNs, two of which are pre-trained, while the other one is self-build and trained, and human functional magnetic resonance imaging (fMRI) brain data to understand how different model architectures and the human brain represent information. By comparing the representations of different DNN architectures, we hope to see which model represents information most similar to humans. To visualize and compare representations of models and human brains during early, middle and late processing stages, we make use of Representational Similarity Analysis (RSA; Kriegeskorte, Mur, and Bandettini, 2008), an effective and widely used tool to compare representations of content across different measurement techniques.

Parts of our analysis are based on a study by Kietzmann, Spoerer, et al., 2019, who showed that recurrence in DNNs is required to capture the dynamic in representations of visual content in the human visual system across different cortical regions. While our research question is not something that has never been tackled before (e.g., Güçlü and van Gerven, 2015; Kietzmann, Spoerer, et al., 2019), the main motivation for our project was rather to get our own hands on real brain data and implement a processing pipeline to compare human brain representations to DNN representations, while understanding the struggles and

decisions that researchers have to make along the way.

# 2 Methods

- *The Methods - how exactly are you solving the problem at hand, how and why does everything work? Consider visualising your architecture and if appropriate important structures from your methods.* The implementation (notice this is different to typical scientific publications):
- *How does your implementation work? Try to explain your implementation choices and walk the reader through your implementation. Consider making use of pseudocode, appropriate visualizations (e.g. for explaining the structure) or even detailing crucial code elements in detail.*

## 2.1 The Data

We made use of the Natural Scenes Dataset (NSD; Allen et al., 2021; http://naturalscenesdataset.org), a freely-available dataset consisting of 73,000 colored natural images from the widely used Microsoft Common Objects in Context (MS-COCO; **10.1007/978-3-319-10602-1˙48** ) image dataset. The NSD dataset comes with whole-brain 7T functional magnetic resonance imaging (fMRI) data with different spatial spaces of participants, recorded during a recognition task. We made use of the 1.8mm resolution. For analyzing and comparing representations between humans and neural networks at different stages of the processing hierarchy, we focused on a subset of the NSD dataset called the *special100* which consists of 100 images that maximally span the semantic space (Allen et al., 2021). We only made use of the fMRI data of subject 1.

## 2.2 fMRI Data Pre-Processing

We made use of fMRI single-trial $\beta$-responses, which are estimations derived through a generalized linear model of fMRI activity that explain the most variance across that trial (betas_fitrhf). Because of storage limitations, we extracted the $\beta$-values from the entire fMRI dataset that correspond to the special100 images and saved them separately.

## 2.3 Regions of Interests

The NSD dataset comes with a number of differently extracted regions of interests (ROIs). We extracted ROIs based on the atlas by Glasser et al., 2016 which defines 180 areas in the human cortex. We focused on early (V1, V2, V3), middle (V4t, LO1, LO2, LO3) and late (IT, PHC, including TE1p, TE2p, FFC, VVC, VMV2, VMV3, PHA1, PHA2, PHA3) stages in the visual hierarchy as in Kietzmann, Spoerer, et al., 2019.

## 2.4 Representational Similarity Analysis

Representational Similarity Analysis (RSA) is a widely used tool to compare representations across different measurement techniques (Kriegeskorte, Mur, and Bandettini, 2008; Nili et al., 2014b). Measured brain activity from a region of interest (ROI) or activity patterns in a layer of a DNN is translated into a point in a high-dimensional response space where each measured unit (e.g., a neuron or a voxel, or a unit in a DNN, depending on the measurement technique) corresponds to one axis. Different stimuli will elicit different activation patterns, the corresponding vectors will therefore point to different locations within the response space. The pair-wise differences between two vectors can be visualized in a representational dissimilarity matrix (RDM). RDMs can be obtained by measuring from different brain regions, stages of processing, humans, species, models, hidden layers in DNNs or behaviour. Further comparing the separate RDMs with each other, where each RDM again corresponds to one vector in a high-dimensional space of several RDMs recorded from different systems for the same set of stimuli, reveals to what degree information is represented differently.

While there are widely used toolboxes for RSA implementation available, we decided to implement the process ourselves to get a better understanding for the technique. The extracted $\beta$-based fMRI activity for all stimuli (n=100) per ROI were extracted. Based on that, a RDMs for each ROI were calculated using the distance between correlations (1 - Pearson correlation).

A second-order RDM was calculated to compare the different first-order RDMs to inspect the relationship between representations in the brain and DNN representations. To do so, we estimate pairwise correlations using Spearman's rank correlation (Nili et al., 2014b).

## 2.5 Pre-Trained Models

We used two pretrained CNNs provided by the Keras module *tf.keras.applications* (TensorFlow, 2022). From the pre-trained models listed there, we chose a 50-layer residual network (ResNet50) and a 16-layer very deep convolutional network (VGG16), both trained on ImageNet.

We decided on ResNet50, introduced by He et al. in 2015, due to its popularity and its success in large-scale image classification (He et al., 2015b). Furthermore, it is known to be a rather simple CNN that generalizes well on other recognition tasks which is very useful in terms of high performance on the MS-COCO dataset which we are using use of. A representative visualization of a ResNet50 model architecture is shown in Figure 1.

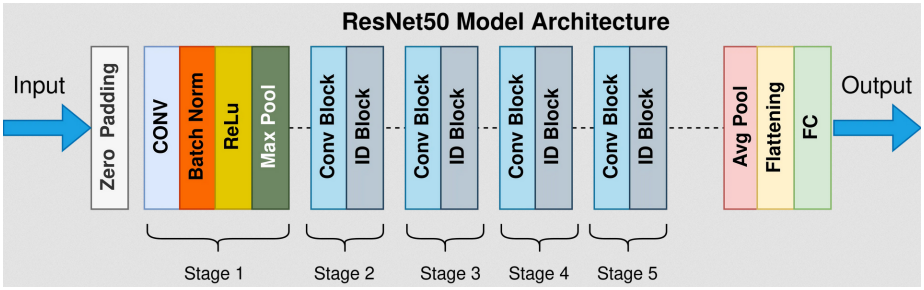


Figure 1: Representative visualization of a standard ResNet50 model architecture (Mukherjee, 2022)

The VGG16 is a CNN with a deeper architecture (Simonyan and Zisserman, 2015; Hassan, 2018). It is has much more convolutional layers and a reduced filter size (3x3). This results into a 92.7% top-5 test accuracy in ImageNet and is perfectly transferable to other image recognition tasks. It was first introduced by K. Simonyan and A. Zisserman in 2015. A representative visualization of a ResNet50 model architecture is shown in Figure 2.

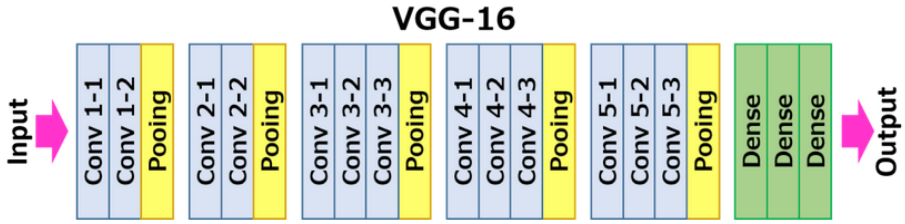


Figure 2: Representative visualization of a standard VGG16 model architecture (Hassan, 2018)

For each pre-trained model we implemented a class that initializes the model with it's pre-trained weights and provides some functions that support our purposes and accesses certain layers. By accessing the layers with our function we get the corresponding layer output activation depending on the input image. The code snipped Listing 1 shows the implementation of the function that returns the layers activation output.

```
def getLayerOut(self, layer_name, data):
    # get the specific layer output
    layer_output = self.base_model.get_layer(layer_name).output
    # build a model on the basis of the original model with the
    output shape of the target layer
    model = Model(self.base_model.input, outputs=layer_output)
    # run predictions
    result = model.predict(data)
```

```
8     return result
```

Listing 1: getLayerOut()-function that returns the defined layer activation based on the input image

## 2.6 Self-Trained Models

In contrast to the chosen pre-trained models, we built a rather small CNN which contains three blocks with two Conv2D layers and one pooling layer each. The first two blocks use a MaxPooling2D-layer while we implemented a GlobalAvPool2D-layer for the last block. As the output layer we defined a dense layer with 91 possible output nodes since our dataset includes 91 different categories. A schematic visualization of our pre-trained model architecture is shown in Figure 3. The corresponding parameter list about all layers can be found in Table 1.

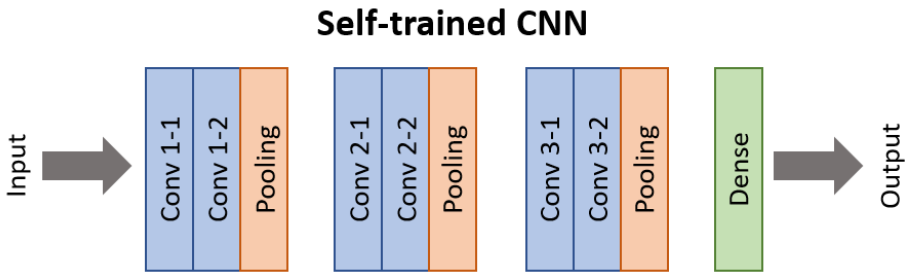


Figure 3: Visualization of our self-trained model architecture.

Table 1: Parameter list of the layers implemented into our self-trained model

| Layer | Filters | Kernel Size | Padding | Activation | Regularizer | Pool Size | Strides |
|---|---|---|---|---|---|---|---|
| Conv 1-1 | 43 | 3 | same | ReLU | L1(0.001) | - | - |
| Conv 1-2 | 43 | 3 | same | ReLU | L1(0.001) | - | - |
| Pooling | - | - | - | - | - | 2 | 2 |
| Conv 2-1 | 86 | 3 | same | ReLU | L1(0.001) | - | - |
| Conv 2-2 | 86 | 3 | same | ReLU | L1(0.001) | - | - |
| Pooling | - | - | - | - | - | 2 | 2 |
| Conv 3-1 | 172 | 3 | same | ReLU | L1(0.001) | - | - |
| Conv 3-2 | 172 | 3 | same | ReLU | L1(0.001) | - | - |
| Pooling | - | - | - | - | - | - | - |
| Dense | - | - | - | Sigmoid | - | - | - |

As an initial learning rate we defined a standard value (learning_rate = 0.001) on which we then applied the Adam optimizer (Adam(learning_rate = learning_rate, decay = 0, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-07)). As a

loss function we chose the BinaryCrossentropy (BinaryCrossentropy(from_logits = True)) since our targets were multi-hot encoded.

We trained the model on a down-scaled set of the MS COCO dataset (image size = 43x43x3) which was saved in a hdf5-file and loaded via google drive due to memory and resource constraints. To encode and build a proper dataset from the hdf5-file we implemented a HDF5-Sequence-Generator that encodes the IDs, the images and the multi-hot encoded targets from the file using the *tf.data.Dataset.from_generator()*-method with a certain structural signature (see Listing 2). We then created image-target-tuples and saved the datasets to the google drive. From there this part didn't need to be run again and we the train and test datasets could be load anytime from the drive which is then preprocessed (normalized, shuffled, batched and prefetched) and put into the model. The model was trained on 118.287 images and tested on 5.000 images.

```
output_signature=({"idx": tf.TensorSpec((),dtype=tf.int32),
                   "image": tf.TensorSpec((43,43,3), dtype=tf.
    uint8),
                    "target": tf.TensorSpec((91,), dtype=tf.
    uint8)})
```

Listing 2: Our output signature to encode the data from the hdf5-fle in a certain structure.

To keep track of the models development, we created summary writers for the test and train phase to track the loss, the precision and the recall of the model. We used precision and recall instead of accuracy because the latter is less suited for multi-label categorization. Precision explains how many categories are correctly predicted and recall checks if all relevant items have been found. Important to know is to have a look at both because a precision of 100% doesn't imply that the predictions are correct. To get an average over the precision and recall the F1-score can be calculated as follows:

$$F1\_score = 2 * ((precision * recall)/(precision + recall)$$

If everything works fine, the F1-score should increase during training.

# 3 Results

- *The results: How did your experiments turn out? Of course training (typically average return vs. training samples) should generally be included, but try to give more details: What exactly happened, can you create an additional experiment to explain what causes your results? Which part of your method causes the results (if you combine multiple approaches?). If appropriate, include a qualitative analysis of your results. Make sure to use appropriate visualization for this section!*

### 3.1 Self-Trained Model

With plotting the losses, the learning performance over time can be visualized and monitored. In this case, the plot of the losses shows that in the very beginning both the training and the test loss are decreasing together (see Fig. 4). But as soon as the second epoch has been trained, it is obvious that the model is overfitting very fast which means that it has learned the training dataset too well.

This phenomenon of overfitting can also be seen in the plots of the precision, the recall and the f1-score of the model (see Fig. 5). These plots show a trade-off between the true positive rate and the positive predictive value for the mode. In this case the training curve is increasing over time while the test curve is first increasing and is saturated or decreases again after a few epochs.

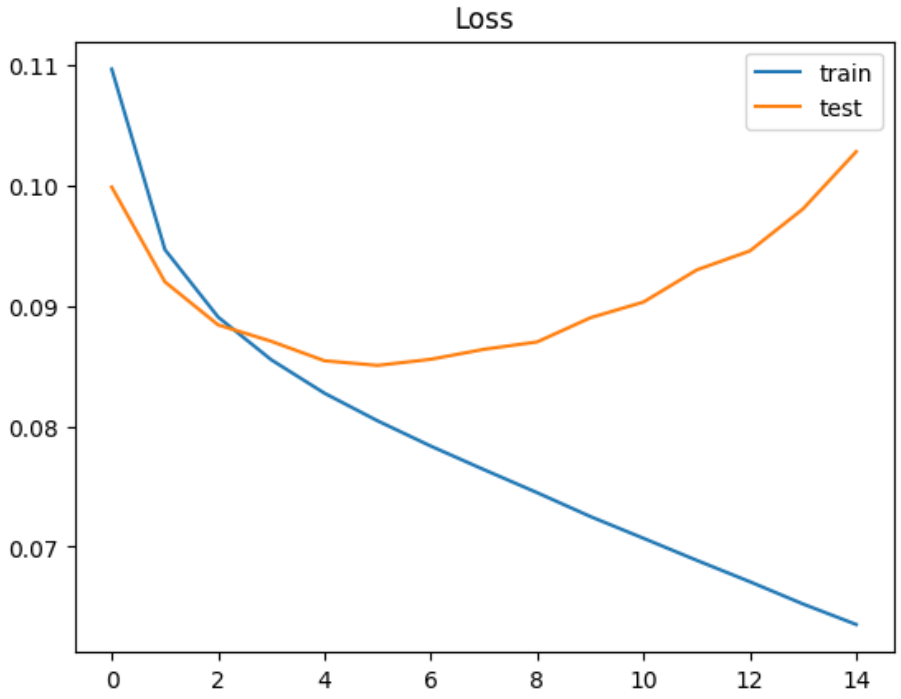


Figure 4: Loss of our self-trained model during training and testing for 15 epochs.

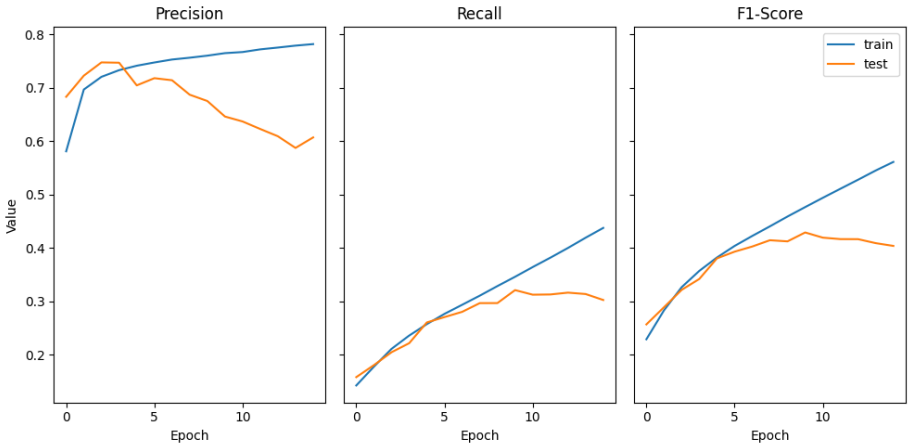


Figure 5: Precision, recall and F1-score as evaluation metrics during training and testing for 15 epochs.

## 3.2 Representational Similarity Analysis

The Representational Dissimilarity Matrices (RDMs) for early, middle and late visual areas show how different the 3 brain regions represent stimuli (see Fig. 6). The RDM of the early visual area shows that the neural activity and corresponding representations between the different images are very different. While the RDM of the middle visual area shows a little more similar representations for some images, the RDM of the late visual area again shows no correlation between the representations of the individual images.

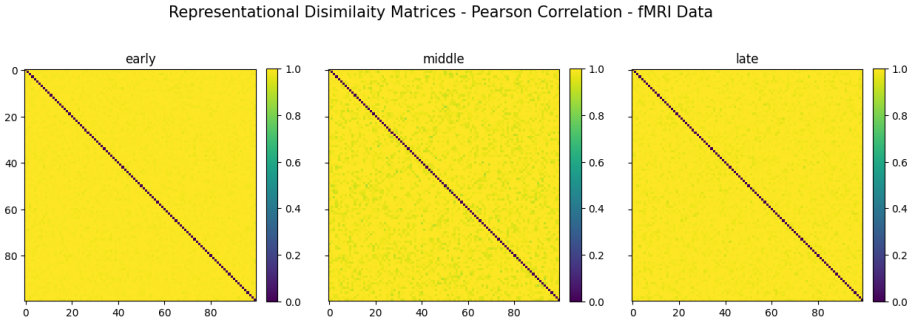


Figure 6: Representational Dissimilarity Matrices (RDMs) for early, middle and late visual areas. X- and y-axes show the numbered NSD special100 images in the order they were shown to subject 1 during the fMRI recording. The dissimilarities were calculated by taking 1-(Pearson Correlation) pairwise.

The RDMs for early, middle and late layers of the ResNet50 show how different the 3 layers represent stimuli (see Fig. 7). The RDM of the early layer shows that the layer activity for the different images are rather similar. The

RDM of the middle layer shows more dissimilar representations than the early layer, while the RDM of the late layer shows even more dissimilarity between the representations of the individual images.

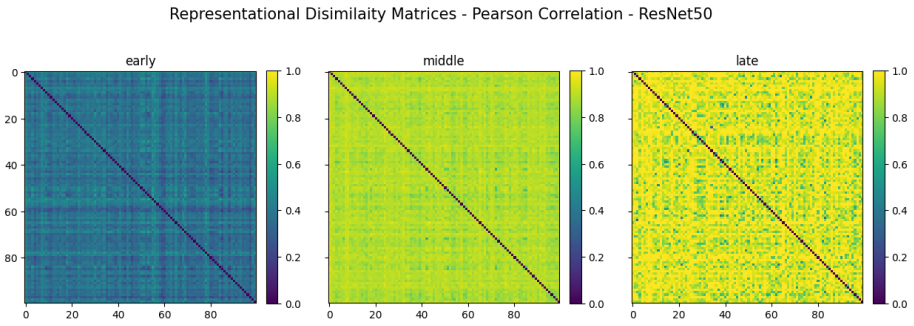


Figure 7: Representational Dissimilarity Matrices (RDMs) for early, middle and late layers of ResNet50. X- and y-axes show the numbered NSD special100 images in the order they were shown to subject 1 during the fMRI recording. The dissimilarities were calculated by taking 1-(Pearson Correlation) pairwise.

The RDM of the early layer of the VGG16 shows that the representations of the different images are moderately similar to each other (see Fig. 8). The RDMs further show that the dissimilarity between representations increases with level of the layer.

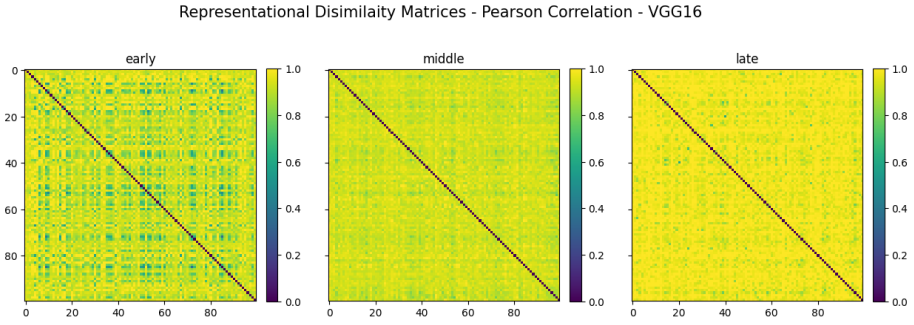


Figure 8: Representational Dissimilarity Matrices (RDMs) for early, middle and late layers of VGG16. X- and y-axes show the numbered NSD special100 images in the order they were shown to subject 1 during the fMRI recording. The dissimilarities were calculated by taking 1-(Pearson Correlation) pairwise.

The RDMs for early, middle and late layers of our self-trained model show that the image representations in the early layer are moderately correlated (see Fig. 9). The dissimilarity between image representations increases in the middle layer, while representations become more similar again in the late layer.

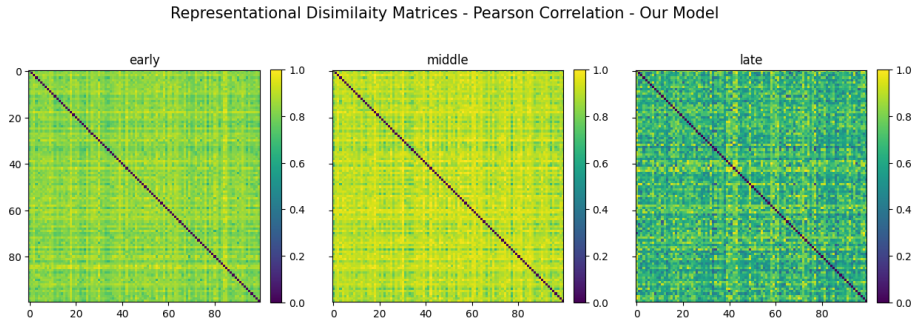


Figure 9: Representational Dissimilarity Matrices (RDMs) for early, middle and late layers of our self-trained model. X- and y-axes show the numbered NSD special100 images in the order they were shown to subject 1 during the fMRI recording. The dissimilarities were calculated by taking 1-(Pearson Correlation) pairwise.

The second-order RDM compares the single RDMs of the different layers/brain areas in the different networks (see Fig. 10). This RDM shows that the representations of early, middle and late visual areas are very dissimilar compared to each other, as well as to the representations in all layers of all other models. However, there seems to be strong, intermediate, as well as some weak dissimilarities between different layers of the different models. For example, we observed a weak dissimilarity (meaning strong similarity) between the middle and later layer of our self-trained model, as well as between the middle layer of VGG16 and the middle layer of ResNet50. Strong dissimilarities (meaning weak similarities) were observed between the early layer of the ResNet50 and the late layer of our self-trained model, as well as between the late and early layers of the ResNet50.


Figure 10: Second-order Representational Dissimilarity Matrix (RDM) comparing early, middle and late layers or visual areas of ResNet50, VGG16, our self-trained model and fMRI data. X- and y-axes show the numbered NSD special100 images in the order they were shown to subject 1 during the fMRI recording. The dissimilarities were calculated by taking the pairwise Spearman rank correlation.

# 4 Discussion

- *Discussion: Put your results and overall project into context: What worked, what did not? What should further be improved? What follow-up questions would you like to ask?*

Because the RDMs of the fMRI data showed that there is no correlation between the representations of the images withing early, middle and late visual areas and across them, we cannot draw any conclusions about how different model architectures represent images similar to the human brain. In the following, we discuss possible reasons for the results we observed, as well as improvements that should be implemented.

### fMRI Data and the NSD Dataset

It was fairly easy to get access to the NDS dataset, including the fMRI data. Reading in and analyzing the fMRI data went well and smooth, also because we could find valuable resources and tutorials online(e.g., https://lukas-snoek.com/NI-edu/getting$_started/about.html$).

### Representational Similarity Analysis

The results obtained from analyzing the fMRI data using our own implemented RSA revealed some interesting findings. When comparing our results with the RDMs obtained through rsatoolbox (**10.1371/journal.pcbi.1003553** ), we observed slight differences. However, other toolboxes seem to have implemented to calculation of the Pearson correlation as we did (e.g., https://github.com/thomastweets/PythonRSA). Further investigations are required to fully understand these discrepancies. The RDMs of the different model layers showed promising results. However, the RDMs of the fMRI data indicated a lack of correlation between the brain activation for different images across early, middle, and late visual areas. It remains unclear whether this is due to the version of the preprocessed fMRI data used. Additionally, we propose that categorizing the NSD special100 images into animate and inanimate images could provide valuable insights into how the brain processes these types of images. Previous research has shown that the brain processes animate and inanimate images differently (Kriegeskorte, Mur, Ruff, et al., 2008), and analyzing the RDMs of these categories could reveal patterns indicating that representations of animate objects are more similar to each other than to representations of inanimate objects. Further investigations are necessary to fully explore these intriguing findings.

### Our Self-Trained Model

We are not completely sure if the evaluation metric is correct, but it's definitely more suitable than accuracy. Its outcome could be more improved by improving the model as we will discuss in a moment.

We tried implementing a deeper model which also looked more promising after the first few epochs, but we were however not able to train it properly due to a shortage of computational resources.

The NSD special100 dataset that we based our analysis on was included in the training data. We know that this is a controversial, but because the special100 images were assigned IDs based on the NSD data set (IDs between 1 and 74000) and our model was trained on MS-COCO with different IDs, we had difficulties finding the corresponding MS-COCO ID for the special100 images. This should definitely be improved.

A problem that could be derived from the loss plot was that the model overfitted very fast what can be a hint that the model learned the training data too well. Therefore the model generalizes bad to new data. To fix this we could have improved the model parameters, like reduce the models capacity, use stronger regularizers or use datasets with higher resolution. But due to the fact that the model was only one part among many other analyzing steps (e.g., fMRI data analysis, incorporating pre-trained models) and we struggled a lot with handling the large amounts of data necessary for training the model with limited computational resources (Google Colab), we did not have the capacity to spend much time on fine tuning and comparing hyperparameters. This could definitely be looked at.

What also could be dicsussed on is that we only used images with a resolution of 43x43 pixels. This was done to not exceed the computational resources and to train the model in a reasonable amount of time. A higher resolution, however, would likely yield better results, because objects would be better visible and the model would be better at extracting relevant features to successfully detect and identify objects.

**Summary**

All in all, we are very happy with the results of this final project. Even though we did not find any striking results, we set up a whole extensive pre-processing pipeline for fMRI data, trained model and worked with pre-trained models. We can say that we really learned a lot.

# References


Allen, E. J., St-Yves, G., Wu, Y., Breedlove, J. L., Dowdle, L. T., Caron, B., Pestilli, F., Charest, I., Hutchinson, J. B., Naselaris, T., & Kay, K. (2021). A massive 7t fmri dataset to bridge cognitive and computational neuroscience. *bioRxiv*. https://doi.org/10.1101/2021.02.22.432340

Glasser, M. F., Coalson, T. S., Robinson, E. C., Hacker, C. D., Harwell, J., Yacoub, E., Ugurbil, K., Andersson, J., Beckmann, C. F., Jenkinson, M., Smith, S. M., & van Essen, D. C. (2016). A multi-modal parcellation of human cerebral cortex., *536*(7615), 171–178. https://doi.org/10.1038/nature18933

Goodale, M. A., & Milner, A. (1992). Separate visual pathways for perception and action. *Trends in Neurosciences*, *15*(1), 20–25. https://doi.org/https://doi.org/10.1016/0166-2236(92)90344-8

Güçlü, U., & van Gerven, M. A. J. (2015). Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *Journal of Neuroscience*, *35*(27), 10005–10014. https://doi.org/10.1523/JNEUROSCI.5023-14.2015

Hassan, M. u. (2018, November 20). *VGG16 - convolutional network for classification and detection*. Retrieved March 31, 2023, from https://neurohive.io/en/popular-networks/vgg16/

He, K., Zhang, X., Ren, S., & Sun, J. (2015a). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, 1026–1034. https://doi.org/10.1109/ICCV.2015.123

He, K., Zhang, X., Ren, S., & Sun, J. (2015b, December 10). Deep residual learning for image recognition. Retrieved March 23, 2023, from http://arxiv.org/abs/1512.03385

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, *160*(1), 106–154. https://doi.org/https://doi.org/10.1113/jphysiol.1962.sp006837

Hung, C. P., Kreiman, G., Poggio, T., & DiCarlo, J. J. (2005). Fast readout of object identity from macaque inferior temporal cortex. *Science*, *310*(5749), 863–866. https://doi.org/10.1126/science.1117593

Kietzmann, T. C., McClure, P., & Kriegeskorte, N. (2019, January 25). Deep Neural Networks in Computational Neuroscience. In *Oxford Research Encyclopedia of Neuroscience*. Oxford University Press. https://doi.org/10.1093/acrefore/9780190264086.013.46

Kietzmann, T. C., Spoerer, C. J., Sörensen, L. K. A., Cichy, R. M., Hauk, O., & Kriegeskorte, N. (2019). Recurrence is required to capture the representational dynamics of the human visual system. *Proceedings of the National Academy of Sciences*, *116*(43), 21854–21863. https://doi.org/10.1073/pnas.1905544116

Kriegeskorte, N. (2015). Deep Neural Networks: A New Framework for Modeling Biological Vision and Brain Information Processing. *Annual Review of Vision Science*, *1*, 417–446. https://doi.org/10.1146/annurev-vision-082114-035447

Kriegeskorte, N., Mur, M., & Bandettini, P. (2008). Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, *2*. Retrieved November 20, 2022, from https://www.frontiersin.org/articles/10.3389/neuro.06.004.2008

Kriegeskorte, N., Mur, M., Ruff, D. A., Kiani, R., Bodurka, J., Esteky, H., Tanaka, K., & Bandettini, P. A. (2008). Matching categorical object representations in inferior temporal cortex of man and monkey. *Neuron*, *60*(6), 1126–1141. https://doi.org/https://doi.org/10.1016/j.neuron.2008.10.043

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *25*, 1097–1105. Retrieved November 26, 2022, from https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context.

In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer vision – eccv 2014* (pp. 740–755). Springer International Publishing.

Lindsay, G. W. (2021). Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future. *Journal of Cognitive Neuroscience*, *33*(10), 2017–2031. https://doi.org/10.1162/jocn_a_01544

Mukherjee, S. (2022, August 18). *The annotated ResNet-50* [Medium]. Retrieved March 31, 2023, from https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758

Nili, H., Wingfield, C., Walther, A., Su, L., Marslen-Wilson, W., & Kriegeskorte, N. (2014a). A toolbox for representational similarity analysis. *PLOS Computational Biology*, *10*(4), 1–11. https://doi.org/10.1371/journal.pcbi.1003553

Nili, H., Wingfield, C., Walther, A., Su, L., Marslen-Wilson, W., & Kriegeskorte, N. (2014b). A Toolbox for Representational Similarity Analysis (A. Prlic, Ed.). *PLoS Computational Biology*, *10*(4), e1003553. https://doi.org/10.1371/journal.pcbi.1003553

Simonyan, K., & Zisserman, A. (2015, April 10). Very deep convolutional networks for large-scale image recognition. Retrieved March 28, 2023, from http://arxiv.org/abs/1409.1556

Tanaka, K. (1996). Inferotemporal cortex and object vision [PMID: 8833438]. *Annual Review of Neuroscience*, *19*(1), 109–139. https://doi.org/10.1146/annurev.ne.19.030196.000545

TensorFlow. (2022). *Module: Tf.keras.applications*. Retrieved March 28, 2023, from https://www.tensorflow.org/api_docs/python/tf/keras/applications

Yamins, D. L. K., & DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, *19*(3), 356–365. https://doi.org/10.1038/nn.4244

Yamins, D. L. K., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., & DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, *111*(23), 8619–8624. https://doi.org/10.1073/pnas.1403112111