# Deep Learning (CS4240) TU Delft – Spring 2024 Group 25

Pepijn Vunderink    Rody Haket    Wim Verschuren

13th April 2024

## 1 Introduction

Written by: Wim
In recent years, few-shot learning has shown promising results for enabling machine learning models to understand and classify new concepts with minimal examples, just like humans can learn from very few instances. Traditional approaches often struggle with this task due to the risk of overfitting when learning from a small dataset. To address this challenge, meta-learning has been explored. This utilises few-shot classification tasks to train models.

Another topic that will be discussed in this blog post is knowledge distillation. Knowledge distillation is a technique used to transfer knowledge from a large, complex model (often referred to as the "teacher") to a smaller, simpler model (known as the "student"). The goal is to enable the student model to achieve performance comparable to the teacher model while being more efficient in terms of computation and memory usage. This is particularly valuable in deploying deep learning models to devices with limited resources, such as mobile phones or embedded systems.

In this blog post we will explore both topics. We will start by touching upon the concepts employed in the paper and introducing the concept of knowledge distillation. After that, we will reproduce the results from the paper [1]. Consequently, we will explore ways to transfer learned knowledge to a smaller Conv4 network.

## 2 Concepts

In this section we will explain some of the concepts that are important for understanding this blog post. Starting of with the Classifier- and Meta-Baseline (section 2.1) and consequently the concept of knowledge distillation in section 2.2.

### 2.1 Classifier- and Meta-Baseline

Written by: Rody and Wim
Few-shot classification using a whole-classification trained model is an existing approach. Moreover, it has resulted in better performance than applying dedicated meta-learning approaches. However, the boundary between whole-classification and meta-learning is not well-known. The

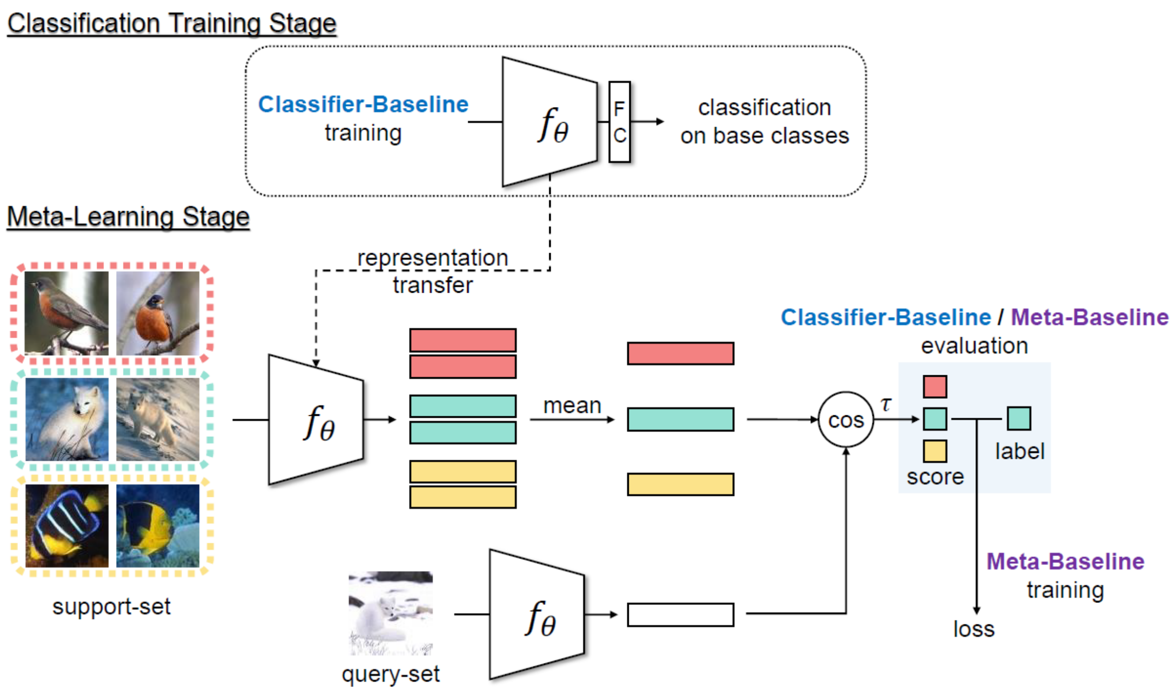

Figure 1: A graphical representation of classifier and meta-baseline. Figure taken from [1]

authors of [1] explore this by evaluating two models.

The objective of standard few-shot classification is to learn novel classes $C_{\text{novel}}$ using a few samples. During training, in an $N$-way $K$-shot few shot classification task, the support set contains $N$ classes with $K$ samples per class. The query-set contains $N$ classes with $Q$ samples per class.

The first model proposed by the paper [1] is a whole-classification model called *Classifier-Baseline*. This model is trained like any standard supervised image classification task. The difference lies in the testing stage. During testing, the last fully-connected layer is removed to obtain an encoder $f_\theta$ that maps the input to an embedding. For a few-shot task, the Classifier-Baseline computes the average embedding of support samples for each novel class to serve as their centroids. It then classifies query samples based on the cosine similarity between the query sample's embedding and the centroids of the novel classes.

The second proposed model is called *Meta-Baseline*. It is attained by fine-tuning the Classifier-Baseline using a meta-learning approach. This means that during the meta-learning stage, $N$-way $K$-shot tasks are sampled from the training samples. The loss is obtained by computing the cross-entropy over the predicted probability distribution for each sample in the query-set. This method introduces an additional learnable scalar $\tau$ to scale the cosine similarity before applying the Softmax function during training. The concepts of *Classifier-* and *Meta-Baseline* are depicted more graphically in figure 1.

Using ResNet-12 as the backbone, Classifier-Baseline is trained on miniImageNet. The authors obtain a $58.91 \pm 0.23$ and $77.76 \pm 0.17$ average 5-way accuracy (%) with 95% confidence interval on both 1-shot and 5-shot tasks, respectively. On miniImageNet, the Meta-Baseline obtains $63.17 \pm 0.23$ and $79.26 \pm 0.17$ average 5-way accuracy (%) with 95% confidence interval, on the 1-shot and 5-shot tasks, respectively.

### 2.2 Knowledge distillation

Written by: Wim
Knowledge distillation is a technique used in machine learning to transfer knowledge from a larger, more complex model (often known as the "teacher") to a smaller model (referred to as the "student" model). The goal is to enable the student model to achieve performance comparable to the teacher model while being more efficient in terms of computation and memory usage. This is particularly valuable in deploying deep learning models to devices with limited resources, such as mobile phones or embedded systems.

The steps to train a teacher-student model are as follows:

1. Training the teacher model

   The teacher model, which is usually a larger and complex neural network, is trained on a given task.

2. Transferring Knowledge

   The student model, which is smaller and simpler than the teacher model, is trained not just to predict the hard labels (the actual class labels), but also to mimic the output distributions (soft targets) of the teacher model. In this case, soft targets refer to the probabilities of each class according to the teacher model. The reason for this is that this distribution often contains richer information than hard labels about the relationships between different classes.

## 3 Reproduction

Written by: Rody
We started by cloning the repository and getting it up and running for future tasks. Next, we downloaded the Mini-ImageNet dataset from an alternative source.

Although the repository mentions that the environment should be `Python 3.7`, we created a local environment in `Python 3.11.7` and used the default `Python 3.10` environment in Kaggle. The authors did not supply a `requirements.txt` file. Fortunately, we got away with simply installing the latest version of each required package. After setting up the environment, we were able to locally run `python train_classifier.py --config configs/train_classifier_mini.yaml"`.

We used the existing code accompanying the paper to train a ResNet-12 model on the Mini-ImageNet dataset. Since we do not have access to dedicated GPUs, we copied the existing code to Jupyter notebook files. This allowed us to run the code on the Kaggle platform.

After the necessary setup, we first trained the ResNet-12 model in Kaggle on *GPU P100*. We used the same parameters as in the original paper. That is, the SGD-optimizer with momentum 0.9, the learning rate starts from 0.1 which decays after epoch 90. We trained for 100 epochs with a batch size of 128.
During testing, we used default configuration parameters. We report the results after 10 test epochs. Table 1 shows the Average 5-way accuracy (%) with 95% confidence interval of both the original paper and our reproduction.

Similarly to the original paper, we fine-tuned our baseline model by applying meta learning for both 1-shot and 5-shot tasks. We ran the meta-learning algorithm for 20 epochs, as set in the existing configuration files.

Interestingly, we find that our reproduction of Meta-Baseline has worse average accuracy for both 1-shot and 5-shot tasks. This is the case both in terms of absolute accuracy as well as in relative difference. This could be explained because we have higher average accuracy for both tasks on the Classifier-Baseline. As explained by the original paper, a higher baseline accuracy indicates higher base class generalisation. The novel class generalisation performance suffers as a result. Although the results are comparable, we hypothesise that this is also the case for our results.

| **Method** | **1-shot** | **5-shot** |
|---|---|---|
| Classifier-Baseline | $58.91 \pm 0.23$ | $77.76 \pm 0.17$ |
| Meta-Baseline | $\mathbf{63.17 \pm 0.23}$ | $\mathbf{79.26 \pm 0.17}$ |
| Classifier-Baseline (ours) | $60.43 \pm 0.22$ | $78.37 \pm 0.17$ |
| Meta-Baseline (ours) | $62.21 \pm 0.23$ | $78.61 \pm 0.17$ |

Table 1: Comparison of few-shot learning performance in terms of accuracy (%)

## 4 Applying Knowledge Distillation to Improve Conv4 Results

Written by: Pepijn, Rody

Our external TA, who proposed this project, has developed specialised hardware for running deep neural networks. It can however only fit models with up to 150k parameters. Models used in the paper, such as ResNet-12 (8M parameters), are too large to fit on this hardware. We are therefore interested to see if we can transfer the knowledge from these larger models to a smaller one (<150k parameters) that would theoretically fit on the hardware.

In this section we investigate how we can transfer the knowledge from the Classifier-Baseline model – which has ResNet-12 as backbone –, trained on the `miniImageNet` dataset, to a smaller model with Conv4 as backbone. We will do this using the knowledge distillation technique described earlier.

The paper [1] provides two baselines on the 1-shot and 5-shot tasks achieved by methods that also use a Conv4 network as backbone. The results of these methods are displayed in table 2.

### 4.1 The Initial Knowledge Distillation Attempt

In our first attempt to investigate if applying knowledge distillation improves the accuracy of few-shot tasks using a Conv4 as the backbone, we first trained the Conv4-network on an architecture similar to that used in [1]. We used our trained ResNet-12 Network as the Teacher Model. During training, at each epoch we perform model inference on both the student and teacher models. For the student model, we first compute the cross entropy loss. Secondly, we compute the *Kullback-Leibler* divergence between student- and teacher outputs. The final loss is defined as $L_s = \alpha L_{KL} + (1-\alpha)L_{CE}$, where $\alpha$ was set at 0.5. We trained the student model for 100 epochs on the 5-way 5-shot task. This is depicted graphically in figure 2.

We only tested the model's performance on the 5-way, 5-shot task as the approach gave subpar results compared to Matching Networks and Prototypical Networks. The test accuracy was $41.39 \pm 0.17$ after 10 epochs. Here, we had the option to further explore the current approach using, for example, hyperparameter optimisation on the $\alpha$ parameter, or implement a more

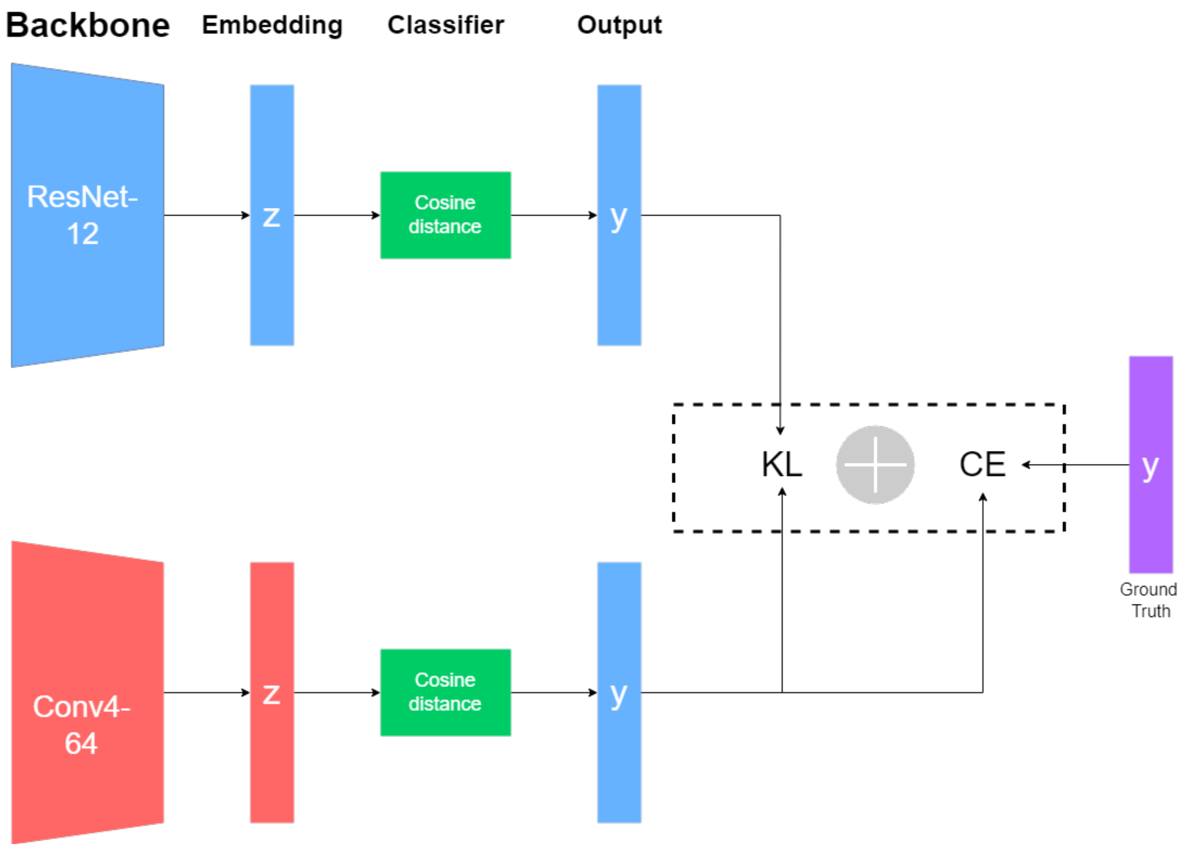

Figure 2: Our initial knowledge distillation approach.

sophisticated algorithm. We agreed that the latter would have more potential. Therefore, we decided to implement additional techniques to improve the application of Knowledge Distillation. In the next section, we describe our implementation and our obtained results in more detail.

## 4.2 Applying Additional Knowledge Distillation Techniques to Improve Conv4 Results

Written by: Pepijn

### 4.2.1 Our Knowledge Distillation Setup

We adopt an approach of knowledge distillation that is very similar to the one used in [4]. While their main contribution lies in clever training of the teacher model, using an additional self-supervised training objective, that improves the knowledge distillation process, we instead use the trained Classifier-Baseline model from [1] that we already have. Concretely, the knowledge distillation approach that we adopt takes the following shape:

- The ResNet-12-based Classifier-Baseline is our teacher model.
- The student model consists of a Conv4-64 model followed by a linear (classifier) layer.
- The loss function, $L = L_{CE} + \alpha_1 L_{KL} + \alpha_2 L_{MSE}$ used for training the student model consists of three terms

  1. $L_{CE}$ this is the Cross-Entropy between ground truth labels and the output of the student network. A hyperparameter $\tau$ – *temperature* – is used to control the softmax over the output of the teacher network.
  2. $L_{KL}$ is the Kullback-Leibler divergence, measuring the difference between (softmax-ed) output of the student and teacher networks.
  3. $L_{MSE}$ is the mean squared error between the embeddings of the encoders of the student ($z_s$) and teacher ($z_t$) networks (these are the output of the Conv4-64 and ResNet-12 parts of the respective models). A linear layer, with trainable parameters, is used to match the embedding size of the teacher (512) to the embedding size of the student (64).

     The distribution of the difference between the teacher embedding and student embedding is calibrated such that it resembles a Gaussian distribution. A hyperparameter $\beta$ is used to control this calibration step.

  This is exactly the loss function used in [4]. Though the paper does not specify that the linear mapping layer for computing $L_{MSE}$ has trainable parameters, we assumed that is what was intended.

Graphically this process is depicted in figure 3. The dataset used for training (`miniImageNet`), along with its split into training and validation sets is exactly the same as the one used for training the teacher model. The hyperparameters used for KD training are $\alpha_1 = 0.3$, $\alpha_2 = 0.5$, $\beta = 0.8$, $\tau = 0.25$, weight decay of 0.0005, batch size 128 and an initial learning rate of 0.025 which is diminished by 0.1 at epoch 70, 100 and 130. We trained for a total of 160 epochs.
We tested the distilled student network on the 5-way 1-shot and 5-way 5-shot tasks, analogously to the testing process in [1].

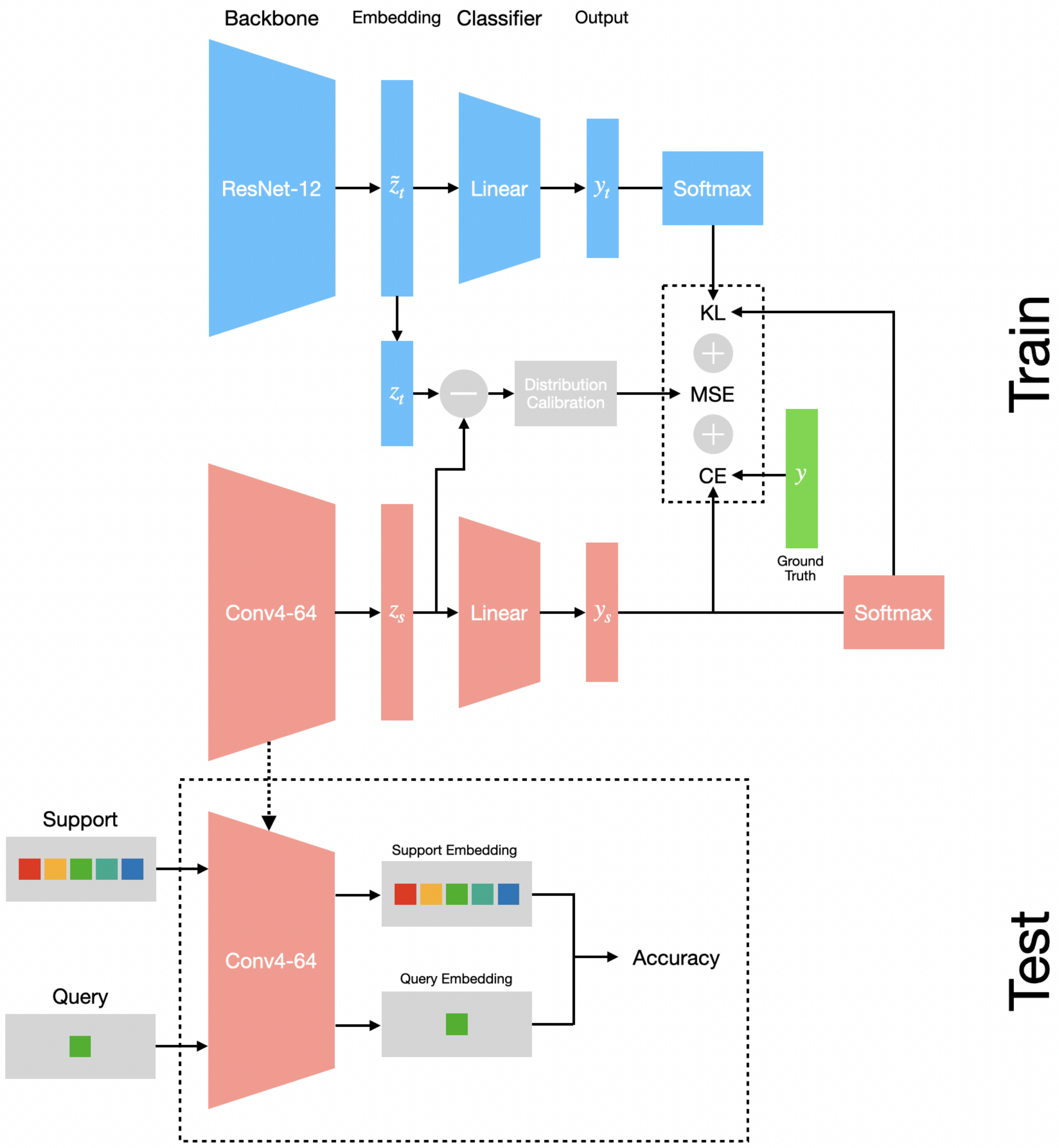

Figure 3: Our knowledge distillation approach.

### 4.2.2 Additional Meta-Learning

In addition to testing the few-shot tasks directly on the distilled Conv4-64 student network, we applied an additional meta-learning step as described in [1]. This additional training step is entirely same as the one applied in the paper, but instead we use our trained student network as the encoder (instead of ResNet-12). This leads to an additional improvement in 1-shot performance, similar to what is seen in the original paper.

## 4.3 Results of Knowledge Distillation on Conv4-64

| **Method** | **1-shot** | **5-shot** |
|---|---|---|
| Matching Networks [3] | $43.56 \pm 0.84$ | $55.31 \pm 0.73$ |
| Prototypical Networks [2] | $48.70 \pm 1.84$ | $63.11 \pm 0.92$ |
| KD (ours) | $48.59 \pm 0.23$ | $66.10 \pm 0.19$ |
| KD + meta (ours) | $\mathbf{51.22 \pm 0.23}$ | $\mathbf{66.61 \pm 0.19}$ |

Table 2: The accuracy (%) results of different methods which all use Conv4 (or Conv4-64) as backbone for few-shot learning.

A summary of the obtained results can be seen in table 2. We observe that, compared to the selected baselines from the meta-learning paper [1], applying knowledge distillation does result in better accuracy on 5-way 5-shot tasks while using Conv4-64 as backbone network. Additionally, applying meta-learning seems to result in a significant improvement on the 5-way 1-shot task.

### 4.3.1 Potential for Improvement

While the performance of our knowledge distillation method outshines the performance obtained by previous methods [2], [3], both of the methods selected for comparison were introduced in 2017 and can no longer be considered state of the art. In fact, the very method proposed in [4], on which we base our approach, outperforms ours by quite a margin. They achieve 57.99% and 73.30% accuracy on the 1-shot and 5-shot tasks respectively. Since their approach mainly differs in the training of the teacher model, a potential angle for future work could be to combine their KD approach with an additional meta-learning training step to potentially further improve performance. Judging by our meta-learning results, especially an improvement in 1-shot performance could be expected.
We speculate that using an even larger model e.g. ResNet-18 or ResNet-50 as teacher model might also lead to better performance, since these significantly outperform the ResNet-12 on few-shot tasks [1].

## References

[1] Yinbo Chen et al. 'Meta-baseline: Exploring simple meta-learning for few-shot learning'. In: *Proceedings of the IEEE/CVF international conference on computer vision.* 2021, pp. 9062–9071.

[2] Jake Snell, Kevin Swersky and Richard Zemel. 'Prototypical networks for few-shot learning'. In: *Advances in neural information processing systems* 30 (2017).

[3] Oriol Vinyals et al. *Matching Networks for One Shot Learning.* 2017. arXiv: `1606.04080 [cs.LG]`.

[4] Bojun Zhou et al. 'Enhancing Few-Shot Learning in Lightweight Models via Dual-Faceted Knowledge Distillation'. In: *Sensors* 24.6 (2024). ISSN: 1424-8220. DOI: `10.3390/s24061815`. URL: `https://www.mdpi.com/1424-8220/24/6/1815`.

## A Division of Tasks

| **Name** | **Task** |
|---|---|
| Wim | • Writing several parts of blogpost: Introduction (1), Concepts (2)<br>• Experimenting with Knowledge Distillation approach |
| Pepijn | • Writing part of the blogpost: Applying KD (4)<br>• Design/implementation of KD setup (section 4.2, 4.3)<br>• Running experiments (section 4.2, 4.3) |
| Rody | • Writing several parts of the blogpost: Concepts (2) Reproduction (3), New variant (4)<br>• Reproduced results on existing codebase (3)<br>• Design, train and test new algorithm variant (4.1) |