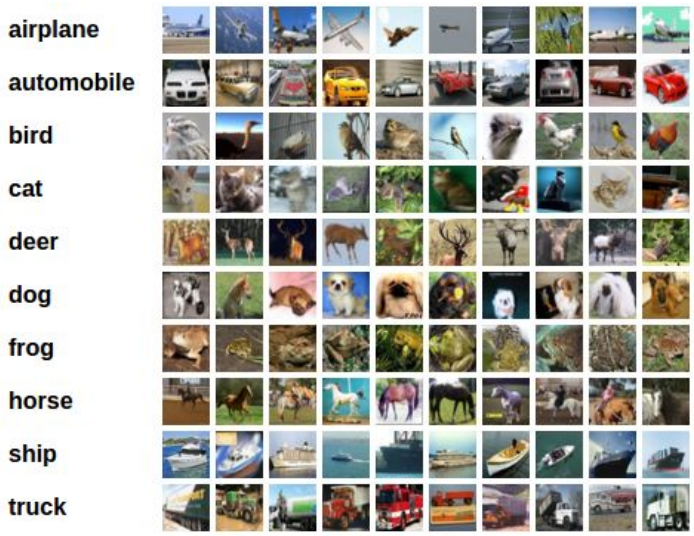
Description

Background

- **Task:** Image classification
- **Goal:**
  - Compare the accuracy of custom networks with different topologies
  - Compare models with
- **Dataset**:
  - to train and compare the accuracy of image classification models.
  - For our experiments, we will use the **CIFAR-10** dataset.
    The **CIFAR-10** dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The classes are completely mutually exclusive.



- **Metrics:**
  - Accuracy
- Loss function:
  - Cross entropy

- **Code of experiments**:
  - https://github.com/MaksTarnavskyi/CompVisionExperiments

Hypotheses/Research Questions/Ideas

For our experiments, we will first try to create from scratch neural networks, try different compositions of layers, layer size, activation functions. For the second block of experiments, we try to finetune pretrained on image-net neural networks and compare them.

Experiment steps

1. Load CIFAR-10 data
2. Create a train, validation, and test data loaders.
   For train/validation, we use split 90/10 of training part
3. Setup model architecture (own from scratch, or model from torchvision)
4. Train model on N epochs
5. Validate during training and save logs
6. Test model performance accuracy on the test set
7. Visualize training loss and Distribution of accuracy between classes

Steps for transfer learning

1. Initialize the pre-trained model
2. Reshape the final layer(s) to have the same number of outputs as the number of classes in our dataset (10 outputs)
3. Define for the optimization algorithm which parameters we want to update during training
4. Run the training step

Plan

Part 1

- Compare the performance of custom models trained using optimizers SGD, Adam, RMSprop

Part 2

- Compare the performance of custom models with and without using data augmentation

Part 3

- Compare the performance of the custom model and VGG-16 model (with and without pretrained weights initialization)

Finally - gather all results in one table, visualize results, and write a summary.

Experiment journal

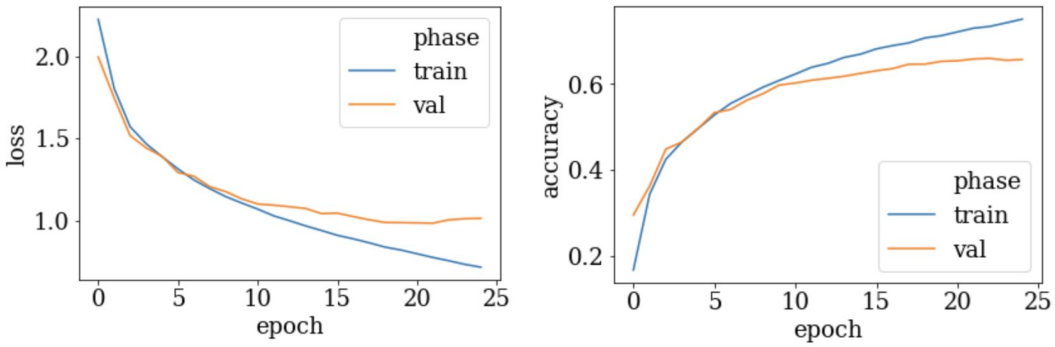
EXP01

Topology of network:

```
Net(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=400, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=10, bias=True)
)
```

Training parameters:

- *Batch size*: 32
- *N_epochs*: 25
- *Optimzer*: SGD(model.parameters(), lr=0.001, momentum=0.9)
- *Criterion*: CrossEntropyLoss
- *Data augmentation*: False



We see that in the end, we have slight overfitting.

Results:

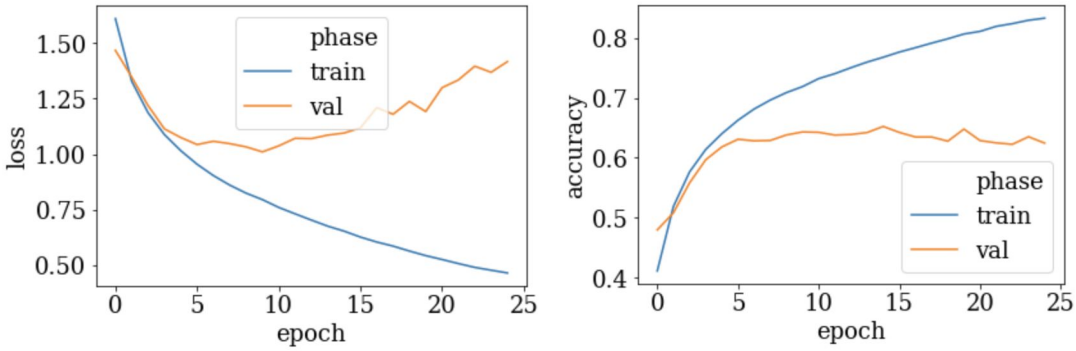
- Best epoch: **22**
- Accuracy of the network on the 10000 test images: **65.5 %**
- Accuracy per class:

	class	Accuracy, %
0	plane	50.93%
1	car	81.2%
2	bird	61.54%
3	cat	44.03%
4	deer	61.16%
5	dog	57.81%
6	frog	77.69%
7	horse	71.2%
8	ship	78.81%
9	truck	60.33%

EXP02

In this experiment, we want to use for our **Net** network the optimizer **Adam**
The topology of the network: The same as in EXP01
Training parameters:

- *Batch size*: 32
- *N_epochs*: 25
- *Optimizer*: Adam(model.parameters(),lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0, amsgrad=False)
- *Criterion*: CrossEntropyLoss
- *Data augmentation*: False



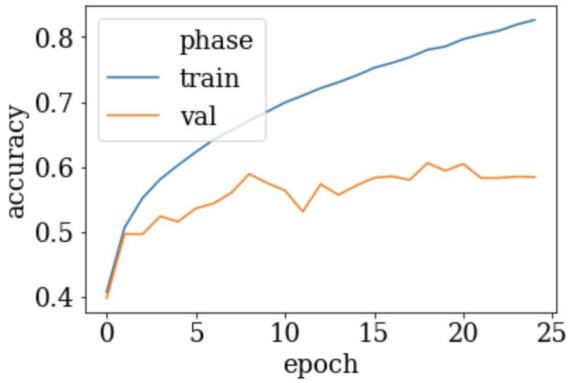
As we see from plots, we have overfitting.
Results:

- Best epoch: **14**
- Accuracy of the network on the 10000 test images: **63.71 %**
- Accuracy per class:

EXP03

In this experiment, we want to use for our **Net** network the optimizer **RMSProp**
The topology of the network: The same as in EXP01
Training parameters:

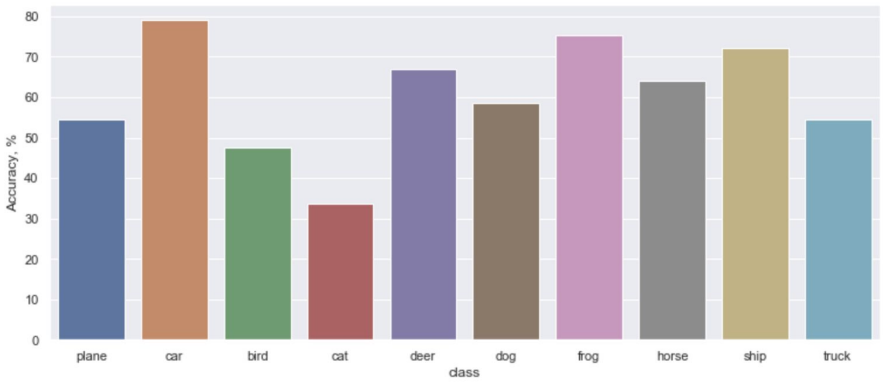
- *Batch size*: 32
- *N_epochs*: 25
- *Optimizer*: RMSprop**(**model.parameters(), lr=0.001, alpha=0.99, eps=1e-08, weight_decay=0, momentum=0, centered=False)
- *Criterion*: CrossEntropyLoss
- *Data augmentation*: False



As we see from plots, we also have overfitting
Results:

- Best epoch: **15**
- Accuracy of the network on the 10000 test images: **62.15 %**
- Accuracy per class:

	class	Accuracy, %
0	plane	54.63%
1	car	78.95%
2	bird	47.55%
3	cat	33.58%
4	deer	66.94%
5	dog	58.59%
6	frog	75.21%
7	horse	64.0%
8	ship	72.03%
9	truck	54.55%



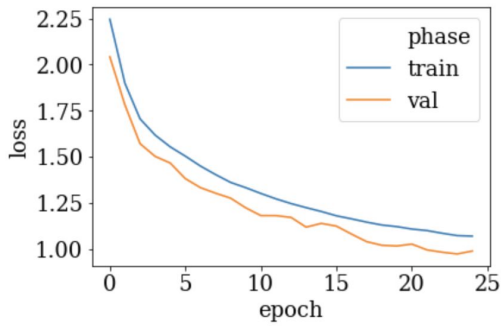
EXP04

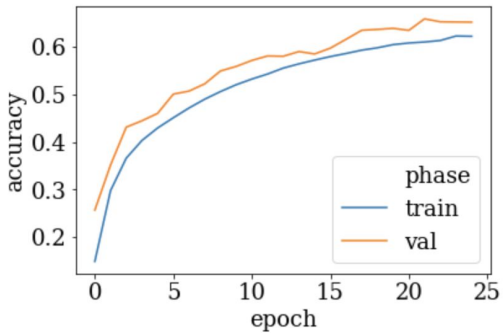
In this experiment, we want to use **data augmentation** for our **Net** network from EXP01

The topology of the network: The same as in EXP01

Training parameters:

- *Batch size*: 32
- *N_epochs*: 25
- *Optimizer*: SGD(model.parameters(), lr=0.001, momentum=0.9)
- *Criterion*: CrossEntropyLoss
- *Data augmentation*:
    - transforms.RandomCrop(32, padding=4),
    - transforms.RandomHorizontalFlip(),

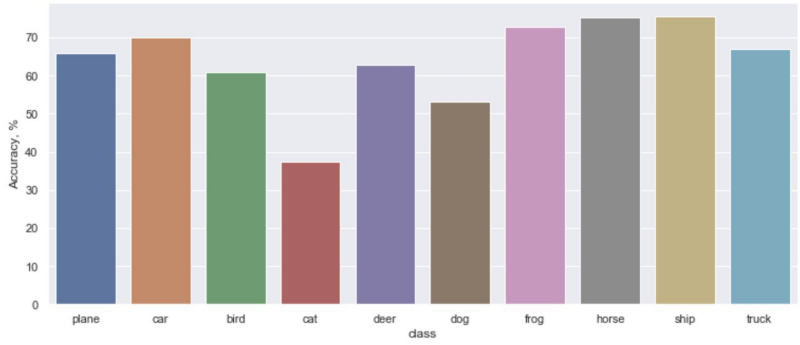




As we see from plots, we don't have overfitting. Training of the model is more regularized. With these additional transformations, it's harder for the model to learn. Probably the model is underfitted.

Results:

- Best epoch: **21**
- Accuracy of the network on the 10000 test images: **64.43 %**
- Accuracy per class:



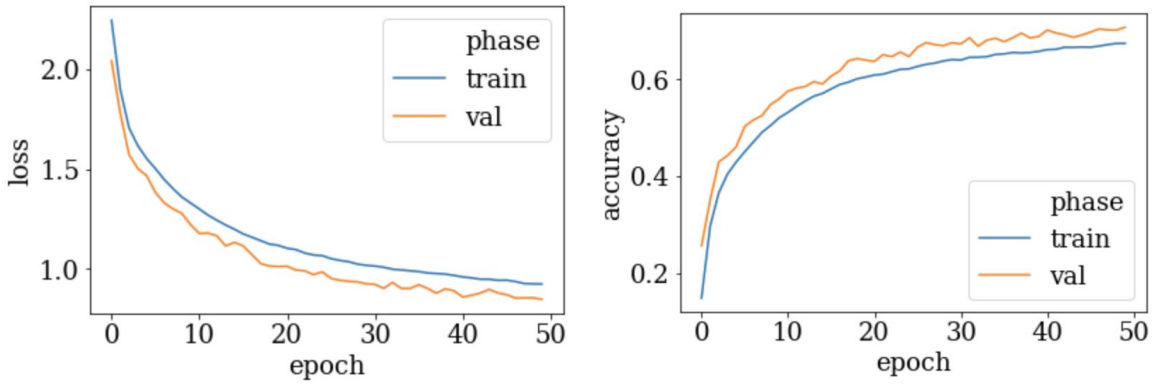
EXP05

In this experiment, we want to use **data augmentation** for our **Net** network from EXP01 with **50** training epochs

The topology of the network: The same as in EXP01

Training parameters:

- *Batch size*: 32
- *N_epochs*: 50
- *Optimizer*: SGD(model.parameters(), lr=0.001, momentum=0.9)
- *Criterion*: CrossEntropyLoss
- *Data augmentation*:
    - transforms.RandomCrop(32, padding=4),
    - transforms.RandomHorizontalFlip(),

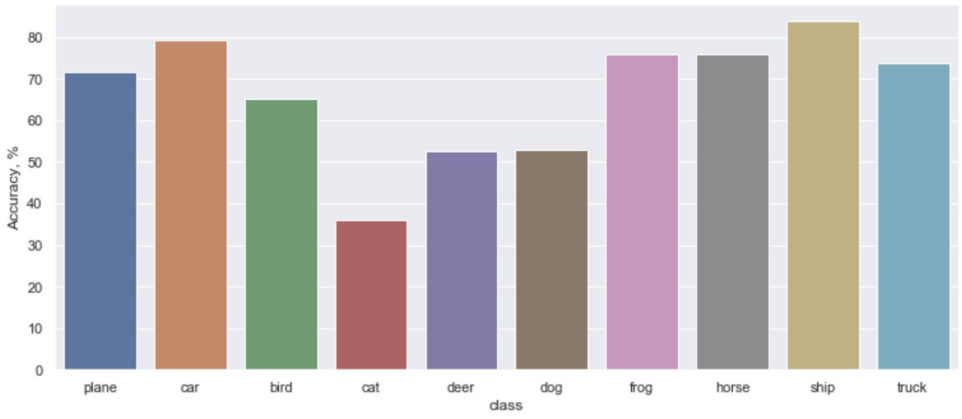


As we see from the plots, we still don't have overfitting. Training of the model is regularized. So we can continue to fit our model and probably will have a better result.

Results:

- Best epoch: **49**
- Accuracy of the network on the 10000 test images: **68.89 %**
- Accuracy per class:

	class	Accuracy, %
0	plane	71.76%
1	car	79.37%
2	bird	65.18%
3	cat	35.9%
4	deer	52.48%
5	dog	52.99%
6	frog	75.91%
7	horse	76.0%
8	ship	83.78%
9	truck	73.73%

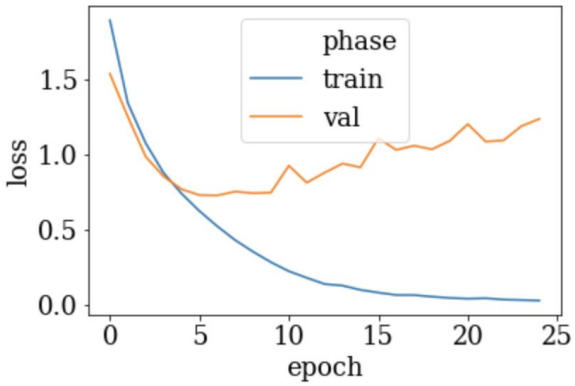


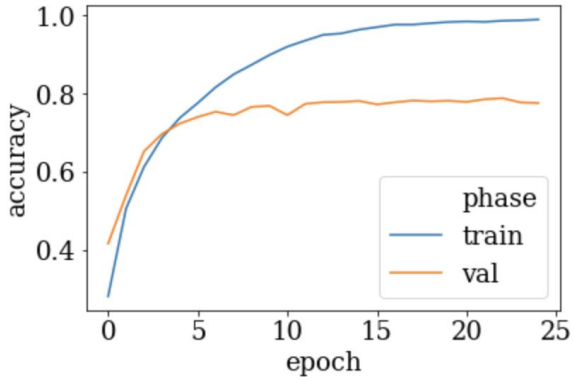
EXP06

In this experiment, we want to train **VGG-16** from scratch (without pre-train weights).
Network: VGG-16

Training parameters:

- *Batch size*: 32
- *N_epochs*: 25
- *Optimzer*: SGD(model.parameters(), lr=0.001, momentum=0.9)
- *Criterion*: CrossEntropyLoss
- *Data augmentation*: False
- *Scheduler*: False
- *Freeze weights*: False

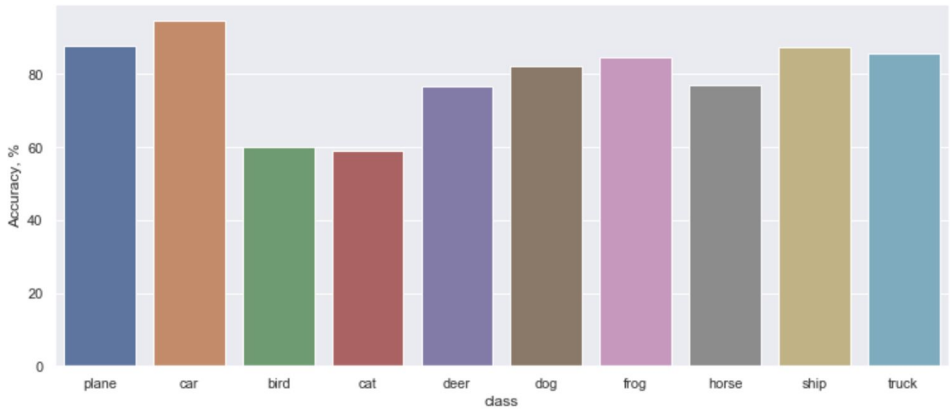




We see that we have overfitting during training. Overall this network has much better accuracy than a simple network from EXP01.
Results:

- Best epoch: **22**
- Accuracy of the network on the 10000 test images: **78.59%**
- Accuracy per class:

	class	Accuracy, %
0	plane	87.79%
1	car	94.53%
2	bird	60.0%
3	cat	58.99%
4	deer	76.7%
5	dog	82.26%
6	frog	84.56%
7	horse	76.92%
8	ship	87.29%
9	truck	85.83%

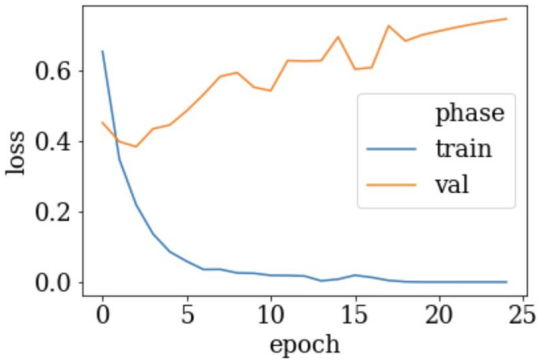


EXP07

In this experiment, we want to train **VGG-16** with **pre-trained** weights on ImageNet.
Network: VGG-16

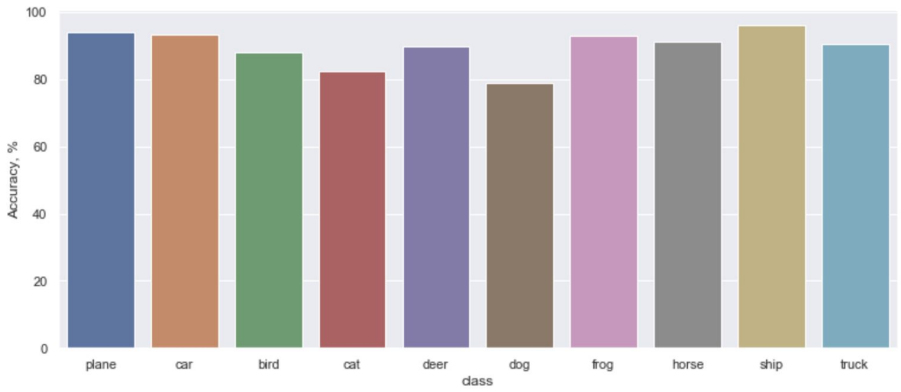
Training parameters:

- *Batch size*: 32
- *N_epochs*: 25
- *Optimzer*: SGD(model.parameters(), lr=0.001, momentum=0.9)
- *Criterion*: CrossEntropyLoss
- *Data augmentation*: False
- *Scheduler*: False
- *Freeze weights*: False



We see that we have huge overfitting during training. Overall this network has much better accuracy than a simple network from EXP01.
Results:

- Best epoch: **23**
- Accuracy of the network on the 10000 test images: **89.38 %**
- Accuracy per class:



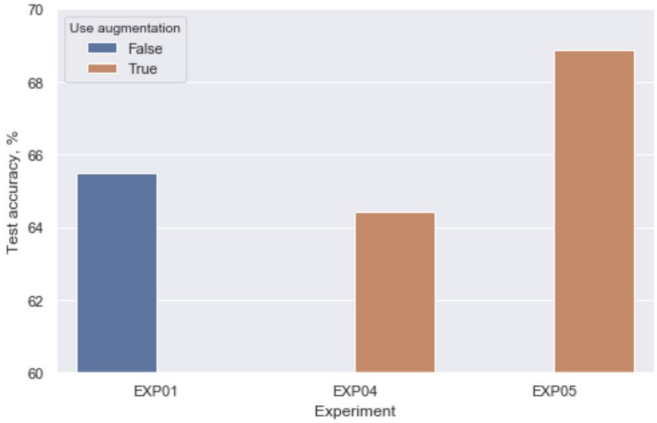
Summary

In the first part, we compared the simple **Net** models that were taught using optimizers SGD, Adam, and RMSProp. All models were training during 25 epochs, and the final models were selected from the checkpoints with the best validation accuracy.

	Experiment	Optimizer	Best epoch	Test accuracy, %
0	EXP01	SGD	22	65.50
1	EXP02	Adam	14	63.71
2	EXP03	RMSProp	15	62.15

All models during training started overfitting. However, the model which used SGD optimizer had the least overfitting and best accuracy result.

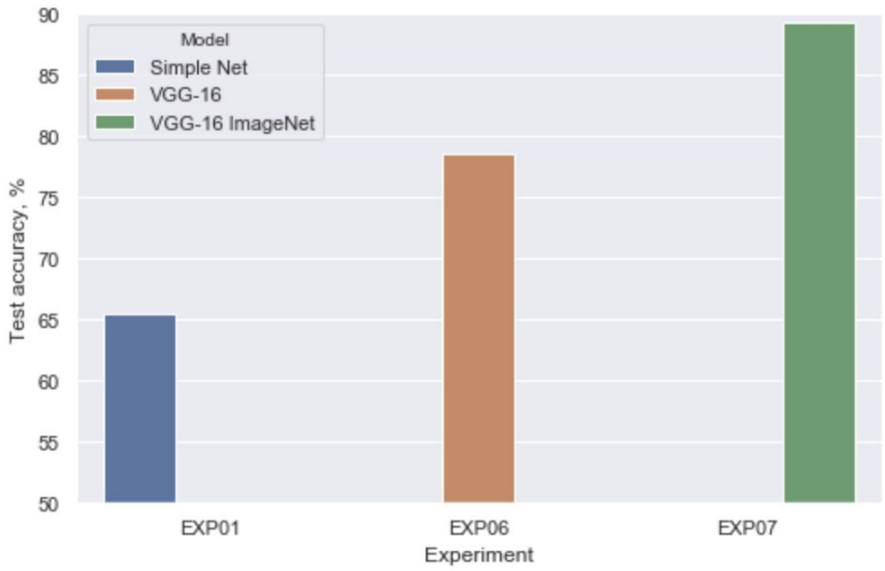
In the second part, we added data augmentation transformations to train dataset loader and taught simple Net with SGD optimizer and compared these results with the results from EXP01.



	Experiment	Use augmentation	Best epoch	Test accuracy, %
0	EXP01	False	22	65.50
1	EXP04	True	21	64.43
2	EXP05	True	49	68.89

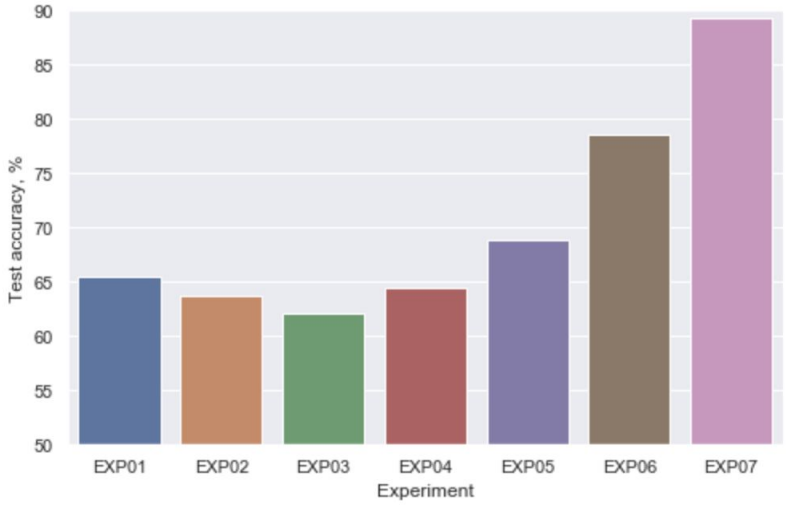
The first model which used data augmentation was underfitted. Its loss values declined more slowly, and the curves did not appear to have been overfitted. Therefore, it was decided to teach the model in more epochs, which significantly improved its result.

In the end, we taught a more complex model **VGG-16** (without and using pre-trained weights) and compared its results with the results obtained in experiment EXP01.



VGG-16 model has better results than simple custom **Net,** and **VGG-16** with initial weights from **ImageNet** has the best results (**89%** accuracy on the test set). This result can be further improved using data augmentation, learning rate scheduler.

Final results between all experiments.



	EXP01	EXP02	EXP03	EXP04	EXP05	EXP06	EXP07
Accuracy	65.5	63.71	62.15	64.43	68.89	78.59	89.38