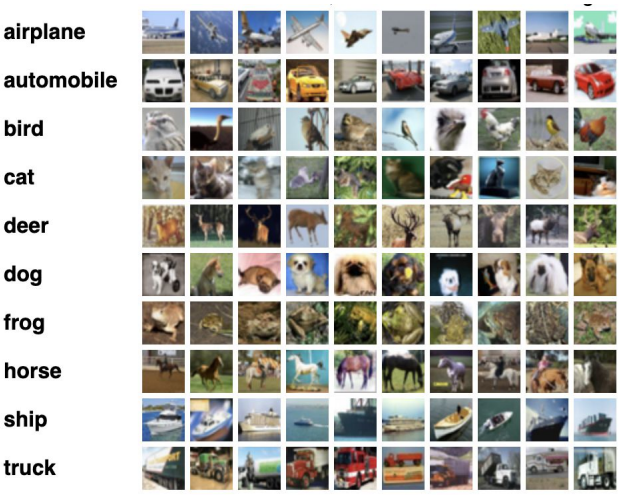
Mini-project report

Deep Learning for Computer Vision

by: Mykola Trokhymovych

Introduction and project description

Problem: image classification on the CIFAR dataset.
Dataset: CIFAR-10, the collection of 60,000 unified resolution, small 32×32 pixel color photographs of objects from 10 different classes:



A random sample of images from CIFAR-10 form https://www.cs.toronto.edu/~kriz/cifar.html

Metric: Accuracy

Background: The project is done as homework for UCU 2020 CV class. The baseline code was given from the course assignment here, and this link with some changes. The main goal is to get as high accuracy results on the testset as possible. All the experiments are done using Google Colab environment and GPU.

Hypotheses:

- We can achieve at least 80% of accuracy with the pretrained models, data augmentation, experimenting with different initializations and optimizations.
- Augmentation, using pretrained models can significantly improve accuracy

Questions/Ideas:

1) How different optimization techniques influence base model performance
2) How different initialization technics influence base model performance
3) Could we significantly improve accuracy using pretrained models.
4) How optimization and initialization influence model results
5) Could we improve model results using data augmentation?
6) How different data augmentation techniques influence accuracy?

Plan:

1) Set a benchmark score using the baseline architecture
2) Experiment with model architecture
3) Experiment with different initializations: constant, Xavier, kaiming
4) Experiment with different optimizations: Adam, RMSprop, SGD
5) Experiment with pre trained imagenet models (Transfer learning).
6) Experiment data augmentation approaches like RandomHorizontalFlip, Resize, RandomRotation.

Experiment journal

[EXP000] Training initial baseline model

Start date	28.12.2020
Finish date	28.12.2020
dataset	cifar-10
history	-

Motivation

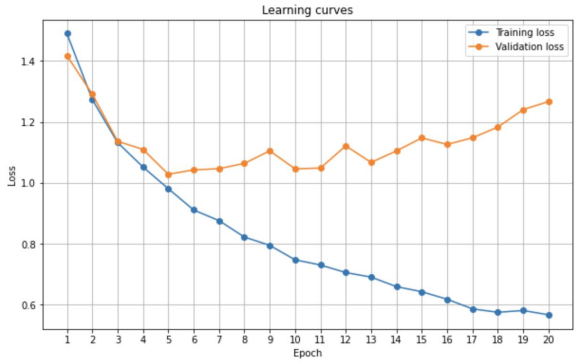
Set a baseline score, that will be the starting point for further experiments. Use basic model from the official tutorial for that.

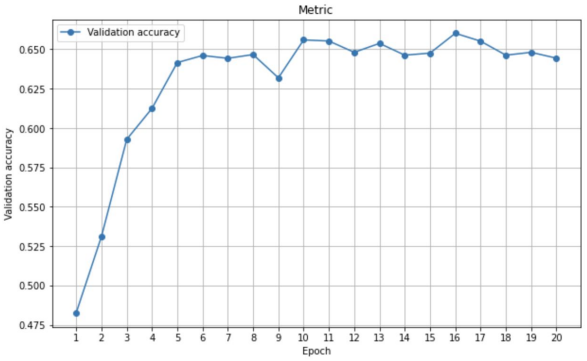
```
Net(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=400, out_features=120, bias=True)
  (fc2): Linear(in_features=120,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

The batch size is 64. Train for 20 epochs. Notebook with experiment: link. The final score is the score of the final model after 20 epochs training.

Results & Deliverables

[accuracy]: 64.44 %





Interpretation & Conclusion

Starting from 6th epoch we started overfitting, due to training/validation loss plots. Should consider early stopping adding, or adding regularization.

Observations

- Baseline is setted to be 65%
- Starting from 7-8th epoch validation loss increase when validation accuracy remains the same, which is not okey

[EXP001, EXP002] Training more advance NN architectures (LeNet, ResNet)

Start date	28.12.2020
Finish date	28.12.2020
dataset	cifar-10
history	linked to exp000 experiment

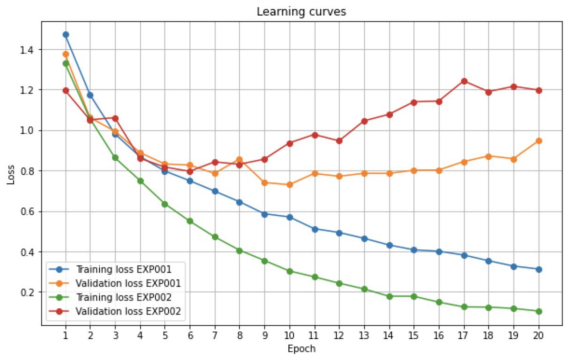
Motivation:

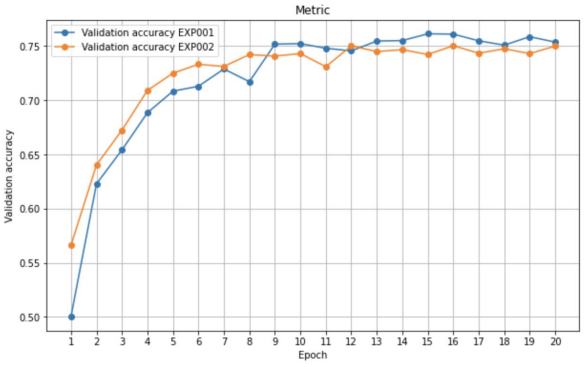
Experiment how more advanced NN architectures can change the score compared to baseline. All the pipeline is the same except model architecture. The experiment is here. The idea is to try out **LeNet** - still basic architecture, but with regularisation (dropout), and some more changes, but still similar to the benchmark. The second experiment is using **EfficientNetB0**, which is a more advanced approach. **LeNet source**, **EfficientNetB0 source**

Results & Deliverables

[accuracy]: 75.37 % (EXP001),
[accuracy]: 75.00 % (EXP002)

Interpretation & Conclusion





EXP001	LeNet architecture
EXP002	EfficientNetB0 architecture

Observations

- Significant improvement of baseline (now we got 75%)
- Both LeNet and EfficientNetB0 have similar results.
- EfficientNetB0 overfitting more comparing to LeNet according to learning curves
- Both models reach plato after 10th epoch (validation accuracy not increasing, validation loss increasing
- The sources for both LeNet and EfficientNetB0 report better results than it was achieved, so we have to look into other parts of the pipeline.

[EXP003, EXP004, EXP005, EXP006] Experiment with initialization using LeNet NN Architecture

Start date	28.12.2020
Finish date	28.12.2020
dataset	cifar-10
history	linked to exp001 experiment

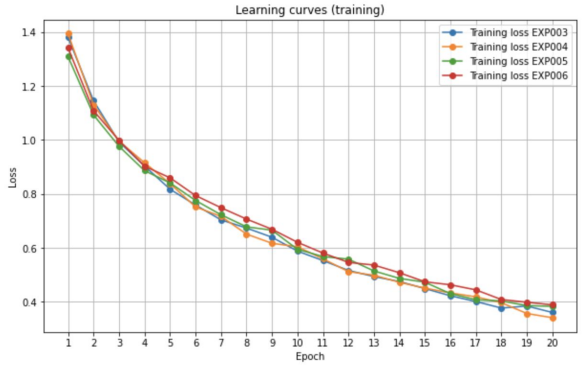
Motivation:

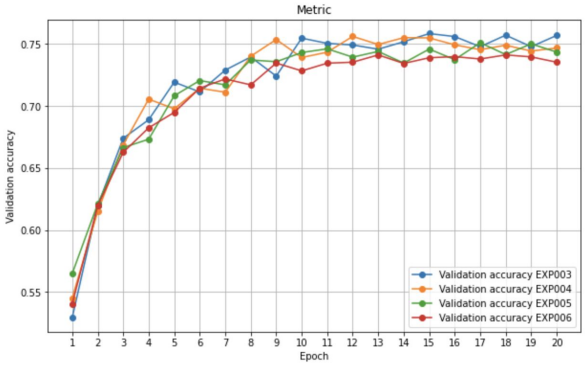
The idea is to compare performance of the models with different initialization techniques: xavier_uniform, xavier_normal, kaiming_uniform, kaiming_normal. The idea is to compare final accuracy and the speed of convergence. All the pipeline and model itself goes from exp001. The experiment code is here.

Results & Deliverables

[accuracy]: 75.70 % (EXP003), [accuracy]: 74.69 % (EXP004),
[accuracy]: 74.36 % (EXP005), [accuracy]: 73.54 % (EXP006)

Interpretation & Conclusion





EXP003	LeNet with xavier_uniform initialization	75.70 %
EXP004	LeNet with xavier_normal initialization	74.69 %
EXP005	LeNet with kaiming_normal initialization	74.36 %
EXP006	LeNet with kaiming_uniform initialization	73.54 %

Observations

- As from the learning curves plot, speed of convergence is almost the same for all initialization techniques.
- The final accuracy is also almost the same. Xavier initialization shows a bit better accuracy results both for normal and uniform variants.
- The best accuracy we have when initialize with xavier_uniform

[EXP007, EXP008, EXP009, EXP010] Experiment with optimizers using LeNet NN architecture

Start date	30.12.2020
Finish date	30.12.2020
dataset	cifar-10
history	linked to exp001, exp003 experiment

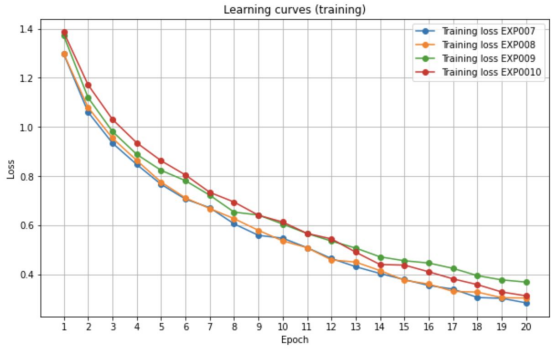
Motivation:

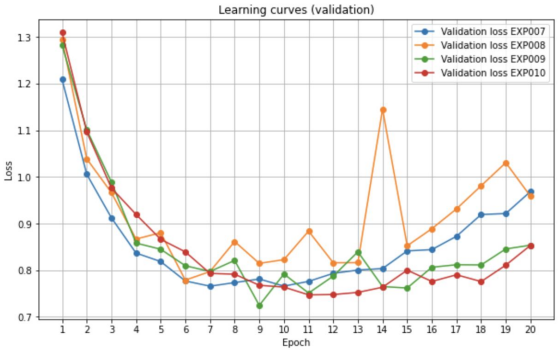
The idea is to compare performance of the models with different optimizers: Adam, RMSprop, SGD. The idea is to compare final accuracy, the speed and smoothness of convergence. All the pipeline and model itself goes from exp001 and I will also use the initialization technique from exp003 The experiment code is here.

Results & Deliverables

[accuracy]: 75.37 % (EXP007), [accuracy]: 74.90 % (EXP008),
[accuracy]: 75.70 % (EXP009), [accuracy]: 75.77 % (EXP010)

Interpretation & Conclusion





EXP007	LeNet with Adam optimizer lr = 0.001	75.37 %
EXP008	LeNet with RMSprop optimizer lr=0.001	74.90 %
EXP009	LeNet with SGD optimizer lr = 0.01, momentum=0.9, nesterov=True (~EXP003)	75.70 %
EXP010	LeNet with Adamax optimizer (default parameters)	75.77 %

Observations

- Models with Adam and RMSProp optimizers have more increasing validation curve, so overfit more than models with SGD and Adamex optimizers.
- RMSProp has the least stability on validation loss
- Adamax is the most stable and has the best results in the end of training

[EXP011, EXP012, EXP013] Transfer learning experiment with resnet18, vgg16, mobilenet_v2 trained on imagenet

Start date	30.12.2020
Finish date	30.12.2020
dataset	cifar-10

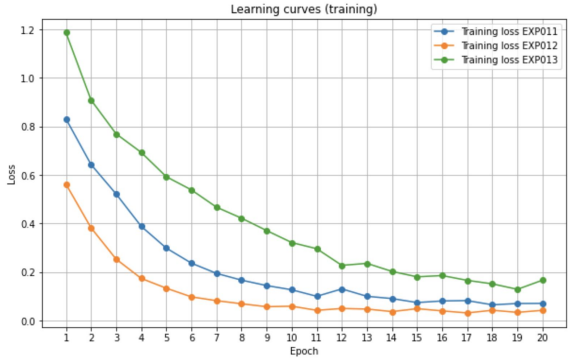
Motivation:

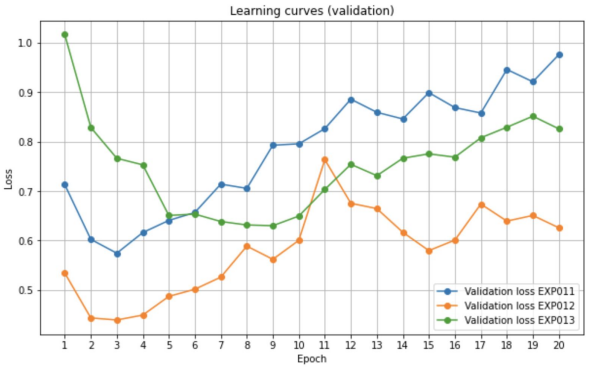
The idea is to use models previosly pretrained on imagenet dataset, change only the last layer to adjust it for our output and finetune models using cifar10. This approach usually gives good result, so I expect a boost in accuracy comparing to previous experiments. The experiments are here. In this experiment I am not freezing convolutions.

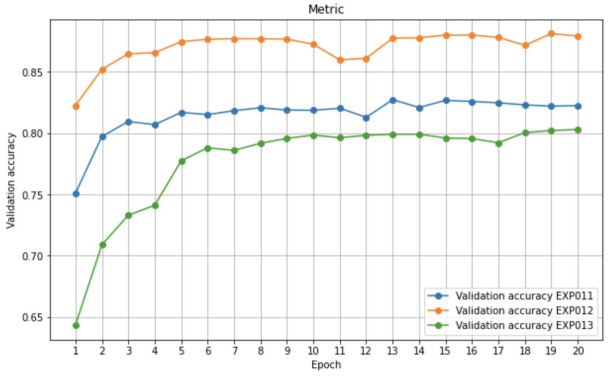
Results & Deliverables

[accuracy]: 82.26 % (EXP011), [accuracy]: 87.93 % (EXP012), [accuracy]: 80.30 % (EXP013)

Interpretation & Conclusion







EXP011	resnet18, Adam optimizer lr = 0.001, not freezed	82.26 %
EXP012	vgg16, Adam optimizer lr = 0.001, not freezed	87.93 %
EXP013	mobilenet_v2, Adam optimizer lr = 0.001, not freezed	80.30 %

Observations

- VGG16 (EXP012) showed the best results with almost 88% accuracy
- Mobilenet_v2 had the worst results, but it was expected as it light model
- Interesting that VGG16 validation loss didn`t have such increasing slope like others,
- Transfer learning really works.

[EXP014, EXP015, EXP016] Transfer learning experiment with resnet18, vgg16, trained on imagenet (freezing pre trained layers)

Start date	30.12.2020
Finish date	30.12.2020
dataset	cifar-10
history	connected with EXP011, EXP012, EXP013

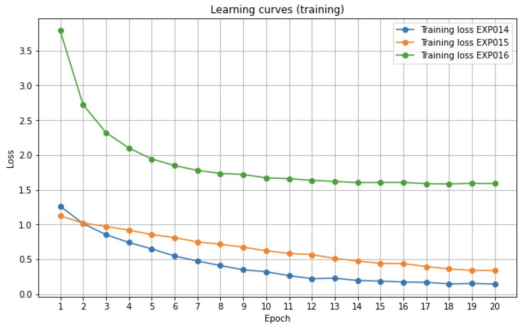
Motivation:

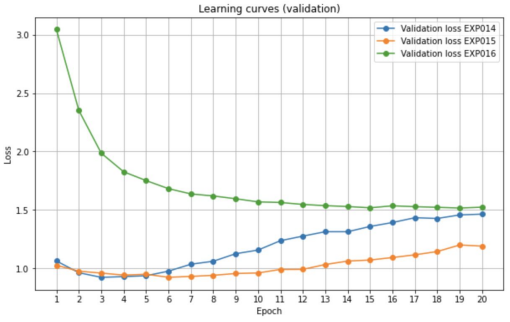
The same as in previous series. The idea is to use models previously pre trained on imagenet dataset, change only the last two layers and finetune only them. The experiments are here. In this experiment I am **freezing** convolutions, so all layers are trainable.

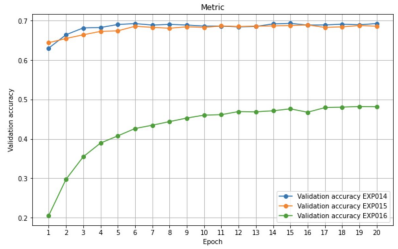
Results & Deliverables

[accuracy]: 69.28 % (EXP014), [accuracy]: 68.61 % (EXP015), [accuracy]: 48.18 % (EXP016)

Interpretation & Conclusion







EXP014	resnet18, Adam optimizer lr = 0.001, only two last are not freezed	69.28 %
EXP015	vgg16, Adam optimizer lr = 0.001, only two last are not freezed	68.61 %
EXP016	mobilenet_v2, Adam optimizer lr = 0.001, only two last are not freezed	48.18 %

Observations

- The results dropped significantly, fine tuning convolutions is important for model
- The speed of training increased comparing with previous experiments
- mobilenet_v2 was almost not training. Also validation loss is different from others.
- vgg16 and resnet18 have almost the same behaviour, but resnet18 overfitting more: validation loss curve has more increasing slope. Interecting that it does not influence validation accuracy - it remains almost the same for both models.

[EXP017, EXP018, EXP019] Transfer learning + adding data augmentation + resize during transformation

Start date	30.12.2020
Finish date	30.12.2020
dataset	cifar-10
history	connected with EXP011, EXP012, EXP013

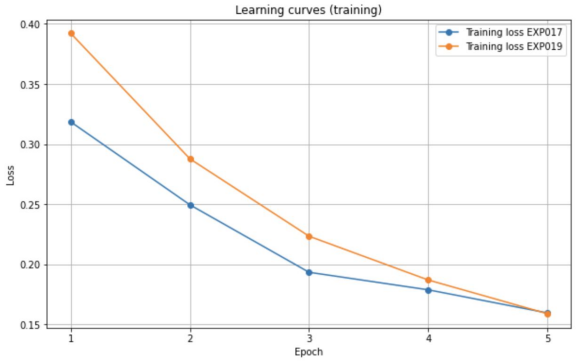
Motivation:

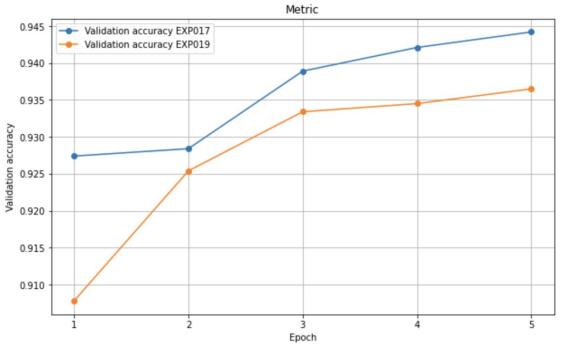
Use pretrained on imagenet models and add various augmentations to train data like RandomHorizontalFlip, Resize, RandomRotation + resize during transformation to match models input size. So, we increase the number of training samples and their diversity. Expect boost in accuracy, but slowdown in training because of more training data. Code is here.

Results & Deliverables

[accuracy]: 94.42 % (EXP017), [accuracy]: 88.58 % (EXP018), [accuracy]: 93.65 % (EXP019)

Interpretation & Conclusion





EXP017	same as EXP011, but 5 epochs only + train set augmentation	94.42 %
EXP018	same as EXP012, after 2 epochs only + train set augmentation. ***not finished***, as was very long running	88.58 %
EXP019	same as EXP013, but 5 epochs only + train set augmentation	93.65 %

Observations

- Data augmentation works great, accuracy increased significantly for two models
- We achieved a cool result (90+% accuracy) even after the first epoch for resnet18 and mobilenet. Both have 10%+ boost in accuracy
- vgg16 was very long running, and didn`t show promising results after first iterations, so I stopped running it. Seems like this augmentation is not working for vgg16.
- Training loss is the same for resnet18 and mobilenet but accuracies are different, resnet is better.

[EXP020] Transfer learning on resnet18 + adding data augmentation + resize during transformation

Start date	30.12.2020
Finish date	30.12.2020
dataset	cifar-10
history	connected with EXP017

Motivation:

I found that there are specific normalization parameters defined for CIFAR10, i will use them in data processing pipeline, so I will change

```
transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
```

to

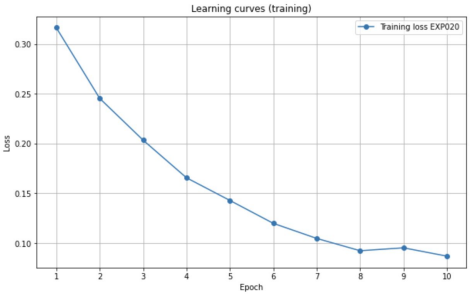
```
transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
```

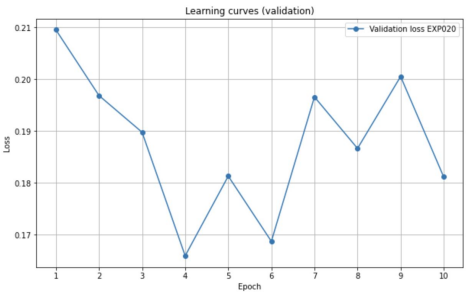
the code is here

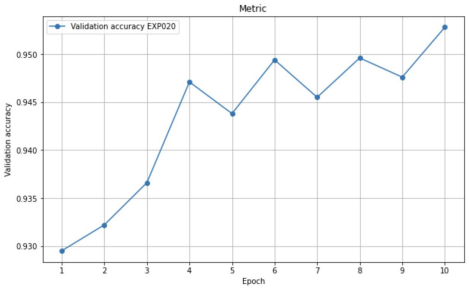
Results & Deliverables

[accuracy]: 95.28 % (EXP020),

Interpretation & Conclusion







EXP017	same as EXP019 + edited data normalization 5+5 epochs	95.28 %

Observations

- This change influenced positively on the first iterations. It achieves both the best validation loss and accuracy on the 4th iteration, but then val loss increases, accuracy decreasing. I decided to train the model for 5 more epochs, and as a result achieved 95.28% accuracy on 10th iteration which is the best current result. But still it is not fair to say that new normalization helped with that as I trained models for more epochs then in case I compared with. Seems like EXP019 should be runned for more epochs also and then compared.
- 95% accuracy achieved.

Summary

General notes:

- The project started with a benchmark score of 64.44% accuracy.
- 20 experiments were performed and documented.
- 95+% accuracy achieved during experiments.
- The best score solution is resnet18 with all layers trainable + data augmentation
- My hypotheses came true: we achieved more than 80% accuracy, pretrained models and data augmentation helped with that.

What was done:

- Experimented with different architectures, optimisation, initialization approaches.
- Used pretrained on imagenet vgg16, resnet18 and mobilenet models - **work**
- Tried out to train only the last layers of pretrained models - **didn't work in my case**
- Tried out to use data augmentation in data processing pipeline - **worked**
- Tried out specific normalization for CIFAR10 - **not sure**

Possible improvements:

- Have some better validation procedure with holdout or cross-validation
- I didn't try to use a scheduler to adjust the training procedure. Using it could also improve results, so such experiments also should be done.
- I was not tuning parameters, like learning rate, some momentums or other configurations. They were picked by common sense, and could be also tuned.
- It is also possible to experiment with different data augmentations, but I am not sure it can significantly improve score
- That would be also great to reproduce some SOTA results, because my scores are somewhere in early 2015: