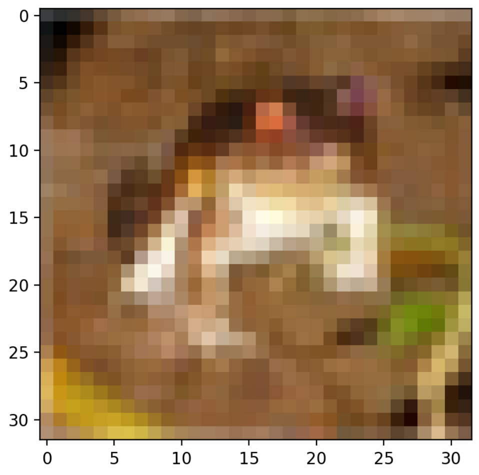
Alessandro De Grandi

Deep Learning Lab Assignment 2: Image Classification Using ConvNets

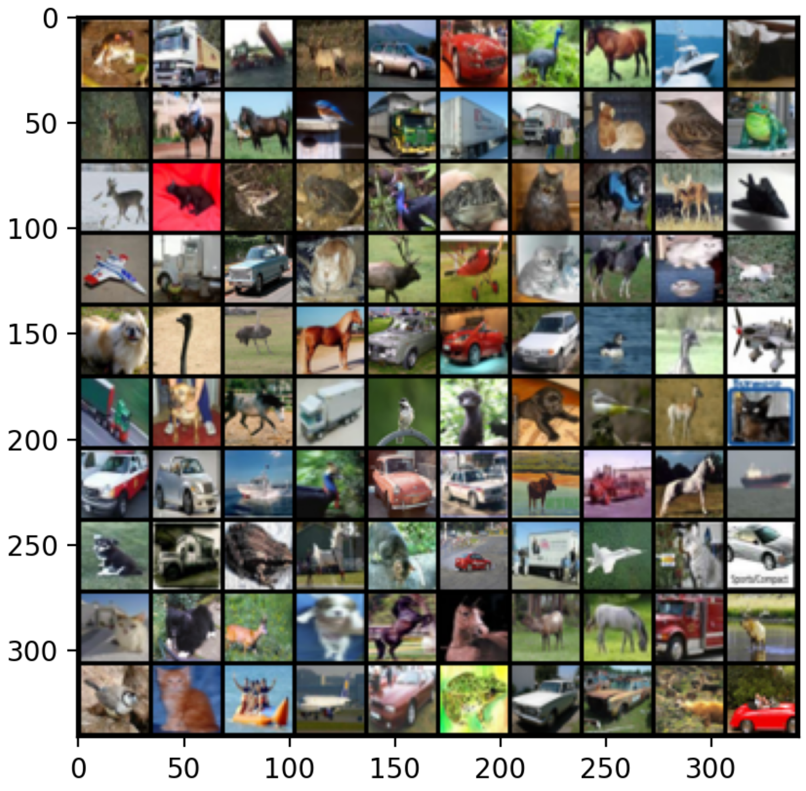
1.1 Dataset

1.1.1. Loaded CIFAR10 dataset, checked number of images = 50000 (each 32 x 32, 3 channels).

First image:



First 100 images displayed on a grid:



1.1.2 Normalize data:

`torchvision.transforms.Normalize(mean, std, inplace=False)`

Parameters:

- mean (*sequence*) – Sequence of means for each channel.
- std (*sequence*) – Sequence of standard deviations for each channel.
- inplace (*bool,optional*) – Bool to make this operation in-place.

Calculated means and standard deviations for each channel and used them as parameters in:

```
transforms.Normalize((mean_r,mean_g,mean_b), (std_r,std_g, std_b))])
```

1.1.3 Split into training and validation:

Used:

```
sampler = torch.utils.data.SubsetRandomSampler(indices)
loader = torch.utils.data.Dataloader(set,batch,sampler,workers)
```

To split the original dataset into 49000 images for training and 1000 images for validation.

1.2 Model:

1.2.1 Implemented the following model:

```
ConvNet(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1))
  (conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1))
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (conv4): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=1600, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=10, bias=True))
```

Using ReLUs activation functions for the Conv2D layers and the first fully connected. The last Layer does not need an activation function because softmax is computed by cross entropy loss.

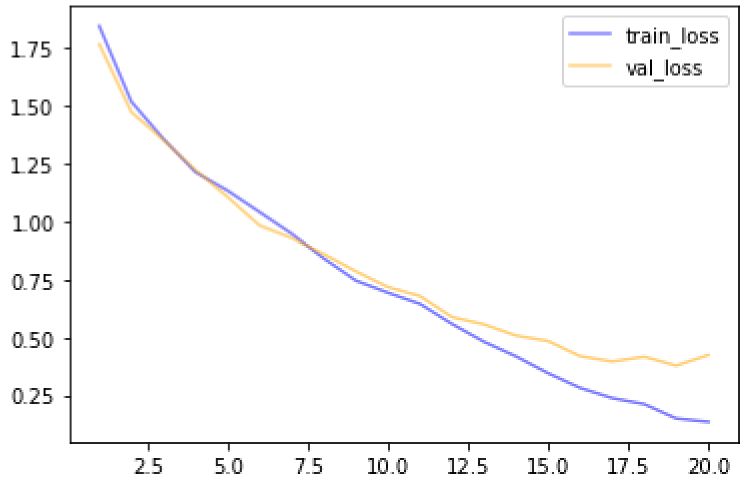
1.3 Training:

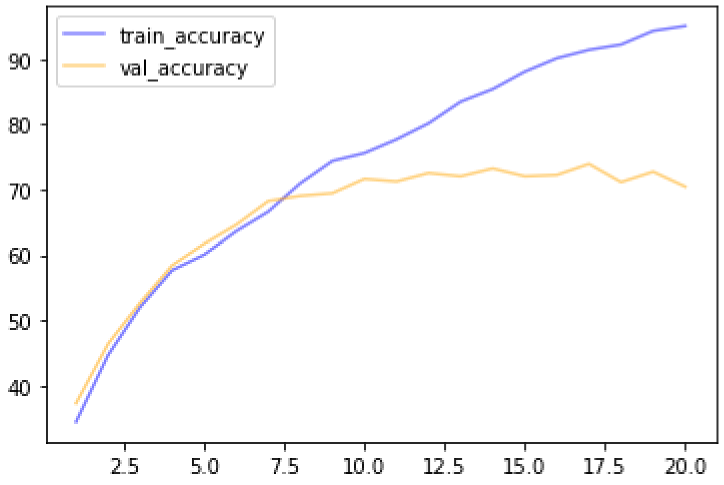
1.3.1 Following the example solution to exercise 4:
Implemented training pipeline, monitoring training loss and accuracy every 200 steps, and validation accuracy after each epoch, also saving the best validation accuracy and the corresponding epoch.

1.3.2 Trained the model with the given hyper parameters

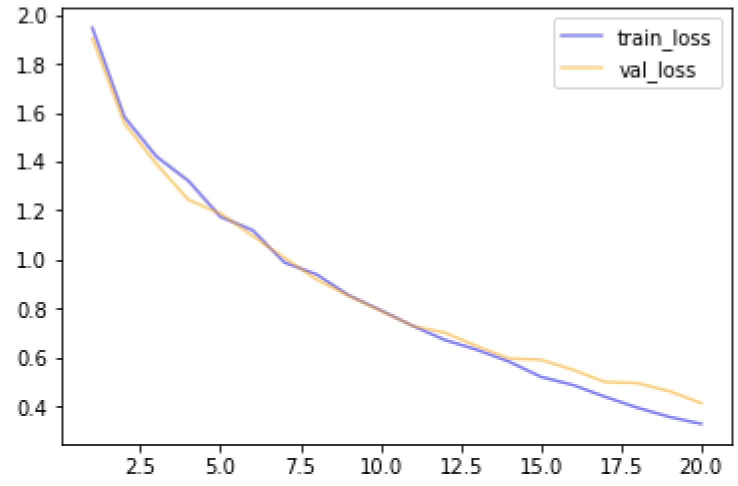
1.3.3 The final validation accuracy is 70% after 20 epochs.

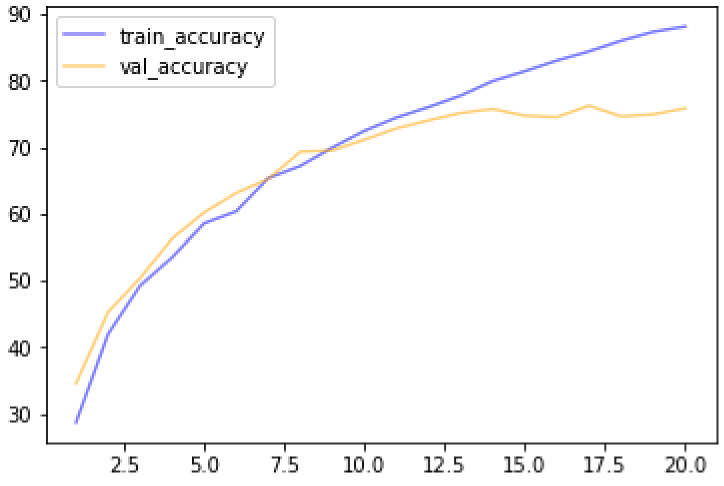
1.3.4 The training and validation losses are very similar to each other for roughly 7 epochs, after that, the model starts overfitting the training data and the difference between the two increases, as the model stops getting better at generalizing.





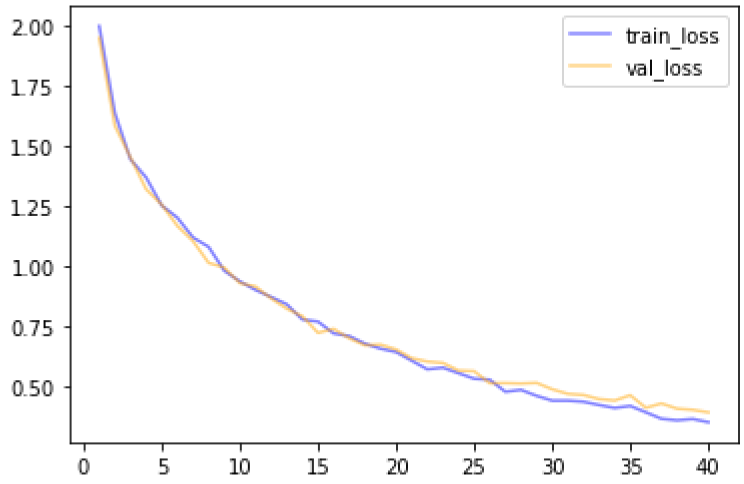
1.3.5 After adding dropout layers with 10% dropout rate the model is doing slightly better at generalizing, with less overfitting. The accuracy has improved to 76%.

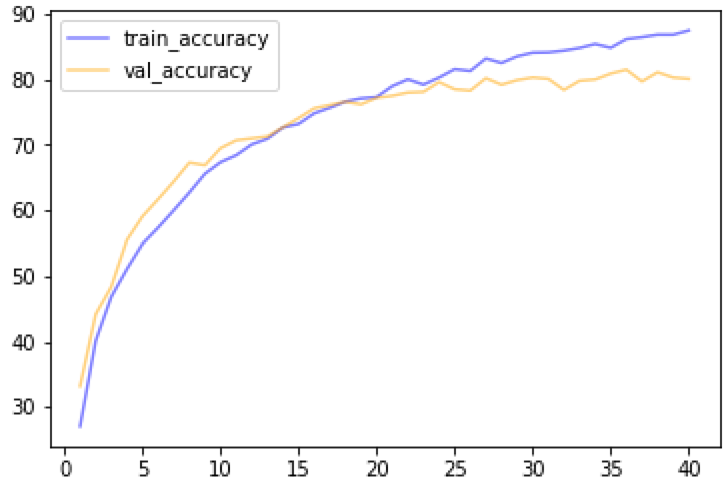




1.3.6 After a number of unsuccessful experiments trying to adjust the dropout rates, I followed the guidelines derived from these two papers:
Improving neural networks by preventing co-adaptation of feature detectors:
https://arxiv.org/pdf/1207.0580.pdf
Analysis on the Dropout Effect in Convolutional Neural Networks:
http://mipal.snu.ac.kr/images/1/16/Dropout_ACCV2016.pdf

So I tried dropout rate of 10% after the first convolutions and pooling, 20% after the second, 50% dropout for the fully connected layer, and 40 epochs.
Reaching a validation accuracy of 80% in epoch 40.
Best validation accuracy 81% in epoch 36.





1.3.7 Test set accuracy is 79%

1.4 Questions

1.4.1. The softmax layer uses a normalized exponential as activation function. It is used as the last layer because it allows to output a probability distribution for the classes. So the class that is assigned the highest probability is used as the prediction.

1.4.2. Like in physics, momentum is used to measure how strongly, in this case the gradient, is moving in a direction; by keeping track of the moving average of past gradients. Intuitively, setting it to non-zero value could help to keep the gradient moving towards the descent direction, and avoid getting redirected by or stuck in local minima, thus converging faster.

1.4.3. The difference is that, in a convolution, the weights are shared across the spatial dimension. This allows the network to learn patterns in spatially related data, like images, where pixels close to each other are not independent.

1.4.4. Convolution can also be used in 1D along a single spatial dimension that is time, eg: sequences of text, time series.

1.4.5. Dropout is a regularization technique used to prevent overfitting. It works by randomly dropping,“ignoring”, a percentage of neurons at each step of the training, thus reducing the model complexity.