TDT4265 - Computer Vision & Deep Learning

Assignment 3 Report - Group 66

> Dionysios Rigatos
> dionysir@stud.ntnu.no

Task 1

Task 1a)

Since the filter is not symmetric and we want to be performing convolution by hand, we will be rotating the kernel twice.

Our new kernel is:

```
1 0 −1
2 0 −2
1 0 −1
```

Let's now perform convolution (only 1 page will be added here as reference for brevity):

$O_{1,2}$

a) $O =$

-11	2	4	-1	7
-24	8	12	-5	12
-19	10	4	-3	15

$\rightarrow O_{3,5}$

$O_{1,1} = 0 \cdot (1+2+1+0-1) + 0 \cdot 2 + 0 \cdot 3 + =$
$= + (-2) \cdot 1 + (-1) \cdot 9 = 0 - 11 = \boxed{-11}$

$O_{1,2} = 0 \cdot (1+0-1) + 2 \cdot 2 + 0 \cdot 1 + (-2) \cdot 2 + =$
$= + 1 \cdot 3 + 0 \cdot 9 + (-1) \cdot 1 = 4 - 4 + 3 - 1 =$
$= \boxed{2}$

$O_{2,3} = 0(1+0-1) + 2 \cdot 1 + 0 \cdot 2 + (-2) 3 + =$
$= 1 \cdot 9 + 0 \cdot 1 + (-1) \cdot 11 = 2 - 6 + 11 + 0 - 1 =$
$= \boxed{4}$

$O_{2,4} = 0(1+0-1) + 2 \cdot 2 + 0 \cdot 3 + (-2) 1 + =$
$= 1 \cdot 1 + 0 \cdot 1 + 4 = 4 + 0 - 2 + 1 - 4 = 0$
$= \boxed{-1}$

$O_{1,5} = 0(1+0+1) + 3 \cdot 2 + 0 \cdot 1 + =$ (2-1)
$= + 1 \cdot 1 + 4 \cdot 0 = \boxed{7}$

(1)

Task 1b)

The answer is (i), the Convolutional Layer. The reason is because in the convolutional layer, each specific kernel extracts a specific feature from each spot of the input image it is applied - and it is applied all over it. Thus, translational shifts do not affect the inference ability of the model as the feature will be extracted no matter where it is in the image.

Task 1c)

We want our output to be $W \times H \times 6$. Since we have $F = 7$ and $S = 1$, we can use the formula (for both H and W):

- $H = (H_{in} - F + 2P)/S + 1 <=> HS - S = H_{in} - F + 2P

  <=> P = (H_{in}S - H_{in} + F - S)/2$

So:

- $P = (1H_{in} - H_{in} + 7 - 1)/2 = (H_{in} - H_{in} + 6)/2 = 3$

That applies for both H and W, so the padding is 3.

Task 1d)

Given $H_{in} = 512$, $W_o = H_o = 508$, $D = 12$, $S = 1$ and $P = 0$, we can use the formula:

- $(H_{in} - F + 2P)/S + 1 = H_{o} <=> (512 - F + 0)/1 + 1 = 508

  <=> 512 - F = 507 <=> F = 5$

So the filter size is 5×5.

Task 1e)

Given, for this layer, $H_{in} = 508$, $F = 2$ and $S = 2$, we can use the formula for pooling layers:

- $H_o = (H_{in} - F)/S + 1 <=> (508 - 2)/2 + 1 = 254$

Equally, $W_o = 254$.

So the spatial dimensions of the pooled feature maps are 254×254.

Task 1f)

We now have $H_{in} = 254$, $F = 3$ and $S = 1$. Using the formula for the convolutional layer:

- $H_o = (H_{in} - F + 2P)/S + 1 = (254 - 3 + 0)/1 + 1 = 252$

Equally, $W_o = 252$.

So the spatial dimensions of the output feature maps are 252×252.

Task 1g)

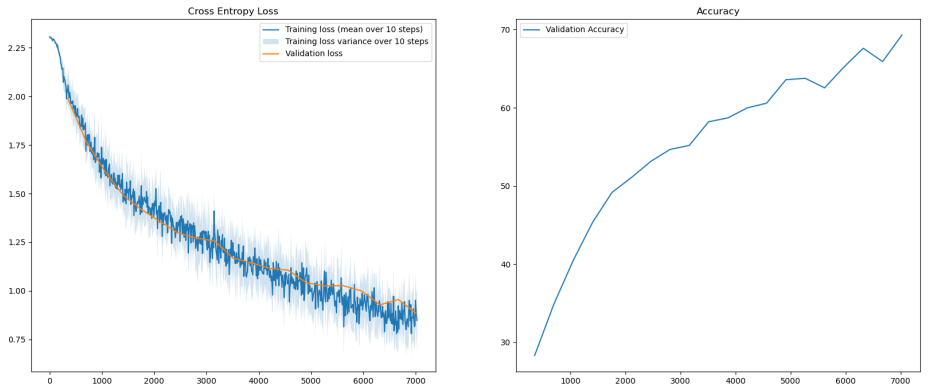
For each layer, we have:

1. $(5 \times 5 \times 3 + 1) \times 32 = 2432$
2. $(5 \times 5 \times 32 + 1) \times 64 = 51264$
3. $(5 \times 5 \times 64 + 1) \times 128 = 204928$
4. Here, the spatial dimensions of the pooled feature map is 4×4. So, the number of parameters is $(4 \times 4 \times 128 + 1) \times 64 = 131136$
5. $(64 + 1) \times 10 = 650$

So, the total number of parameters is
$2432 + 51264 + 204928 + 131136 + 650 = 390410$.

Task 2

Task 2a)



Task 2b)

- **Training Accuracy**: 73.36%
- **Validation Accuracy**: 69.3%
- **Testing Accuracy**: 68.61%

This took 9 epochs.

Task 3

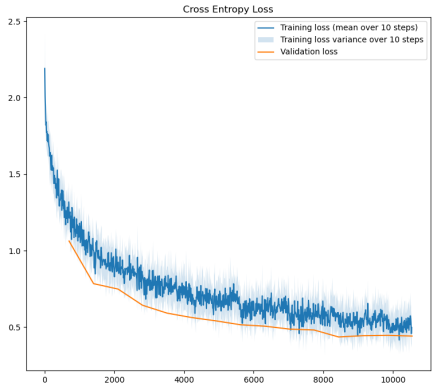
Task 3a)

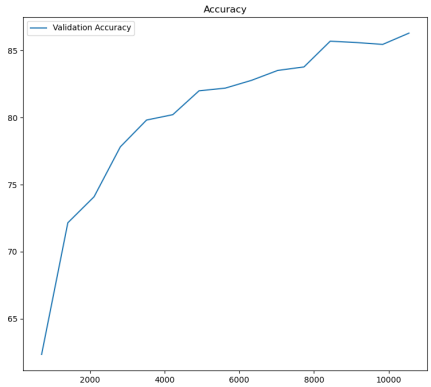
Our over-75% model has the following structure:

```
ImprovedModel(
  (feature_extractor): Sequential(
    (0): Conv2d(3, 32, kernel_size=(5, 5), stride=(1, 1), padding=
(2, 2))
    (1): PReLU(num_parameters=1)
    (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (4): Dropout(p=0.2, inplace=False)
    (5): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1), padding=
(2, 2))
    (6): PReLU(num_parameters=1)
    (7): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (8): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (9): Dropout(p=0.2, inplace=False)
    (10): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2))
    (11): PReLU(num_parameters=1)
    (12): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (13): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (14): Dropout(p=0.1, inplace=False)
    (15): Flatten(start_dim=1, end_dim=-1)
  )
  (classifier): Sequential(
    (0): Linear(in_features=2048, out_features=64, bias=True)
    (1): PReLU(num_parameters=1)
    (2): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (3): Linear(in_features=64, out_features=10, bias=True)
  )
)
```

One can replicate it by running the task3_train_75 notebook. A breakdown of what was used to reach this improvement in accuracy is described in the Task3 section - however a quick note is that the training hyperparameters were not altered.

Task 3b)





- **Training Accuracy**: 81.8%
- **Validation Accuracy**: 81.9%
- **Testing Accuracy**: 76.77%

This took 9 epochs.

Task 3c)

What worked well:

- **Data Augmentation** worked very well. We had to be careful so as to make sure we only augment the train data and not the validation/test data. The model is now able to generalize better as it is learning to distinguish more features in different conditions (i.e. noisy, rotated etc). Specifically:
  - **RandomResizedCrop** was added, which only keeps a random part of the image as a training example.
  - **ColorJitter** was added, which adds color noise in the image.
- **Batch Normalization** helped make sure we converge earlier than usualy by smoothening out the gradient landscape, helping us stay in track with the task requirements. Note: We added BN for both feature extraction and FC inference.
- **Dropout** reduced overfitting during training, as expected, as we are training an ensemble of models with different neurons activated. Only low dropout values seemed to work well - going over 0.3 resulted in a performance loss (probably because we dropped too many parts of features that were relevant).
- **PReLU** activation slightly improved activation, acting as an improvement over LeakyReLU (which helps diminishing gradient), by having a learnable slope.
- **Adam** optimizer slightly improved performance which is expected, as it utilizes adaptive momentum allowing us to converge into a solution much faster, and better minima.

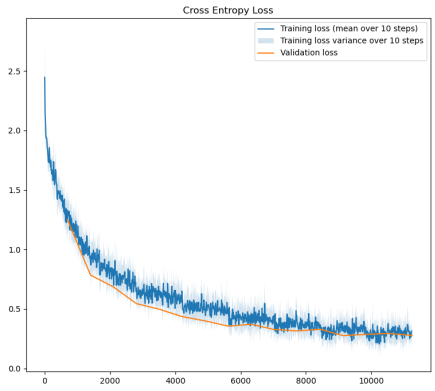
What did **not** work well:

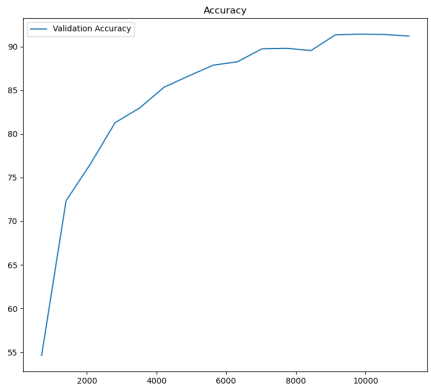
- **Decreasing Filters** lowered the accuracy as expected, as the model was not able to capture enough features off the images and thus not generalizing very well.
- **Decreasing Filter Size** did not work very well either, probably because for this specific dataset, the features are in larger grids.
- **Adjusting batch_size, learning_rate** did not help and the original values performed very well.
- **L2 regularization** ontop of Dropout did not seem to have any effect.

Task 3d)

The golden method seemed to be **Data Augmentation** in combination with Dropout.

Task 3e)





For tis final improvement, here's what I did:

- **Increase number of filters** to 64, which allowed for more features to be captured.
- **Increasing amount of layers**, and specifically applying 2 convolutions before maxpooling as suggested (not replacing Maxpooling!), gave the final push to consistently surpassing 80% accuracy on the test set. More complex patterns are captured in the data and convergence now happens on less than 7-8 epochs! Applying a more sophisticated architecture did indeed help.

- **Training Accuracy**: 90.0%
- **Validation Accuracy**: 89.24%
- **Testing Accuracy**: 84.18%

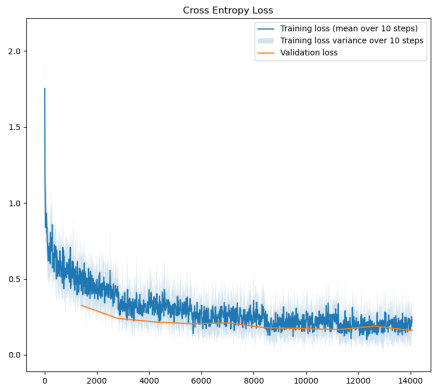
This took 8 epochs.

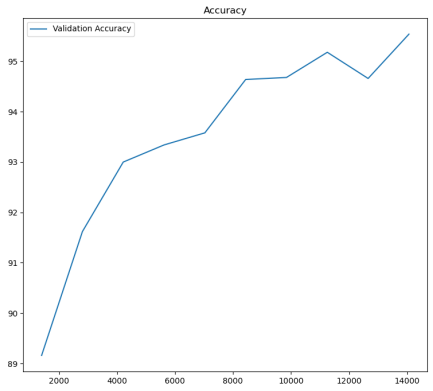
Task 3f)

The model seems to overfit slightly seeing how the training/val accuracy are significantly higher than the testing accuracy. The resutls are still satisfactory, and regularizing further almost always led into the model not meeting the target for the test accuracy.

Task 4

Task 4a)





- **Training Accuracy**: 93.75%
- **Validation Accuracy**: 94.54%
- **Testing Accuracy**: 89.98%

The hyperparameters are as follows:

- **Batch Size** of 32
- **Learning Rate** of 5e10-4.
- **Early Stop** count of 3.
- **Data Augmentation** identical to Task3.
- **Optimizer** Adam.

and the original image size of 224x224.

When training with the ResNet, we perform image normalization, as specified in the imagenet-normalization-implementation link, with values:

- **Mean** of [0.485, 0.456, 0.406]
- **Standard Deviation** of [0.229, 0.224, 0.225]