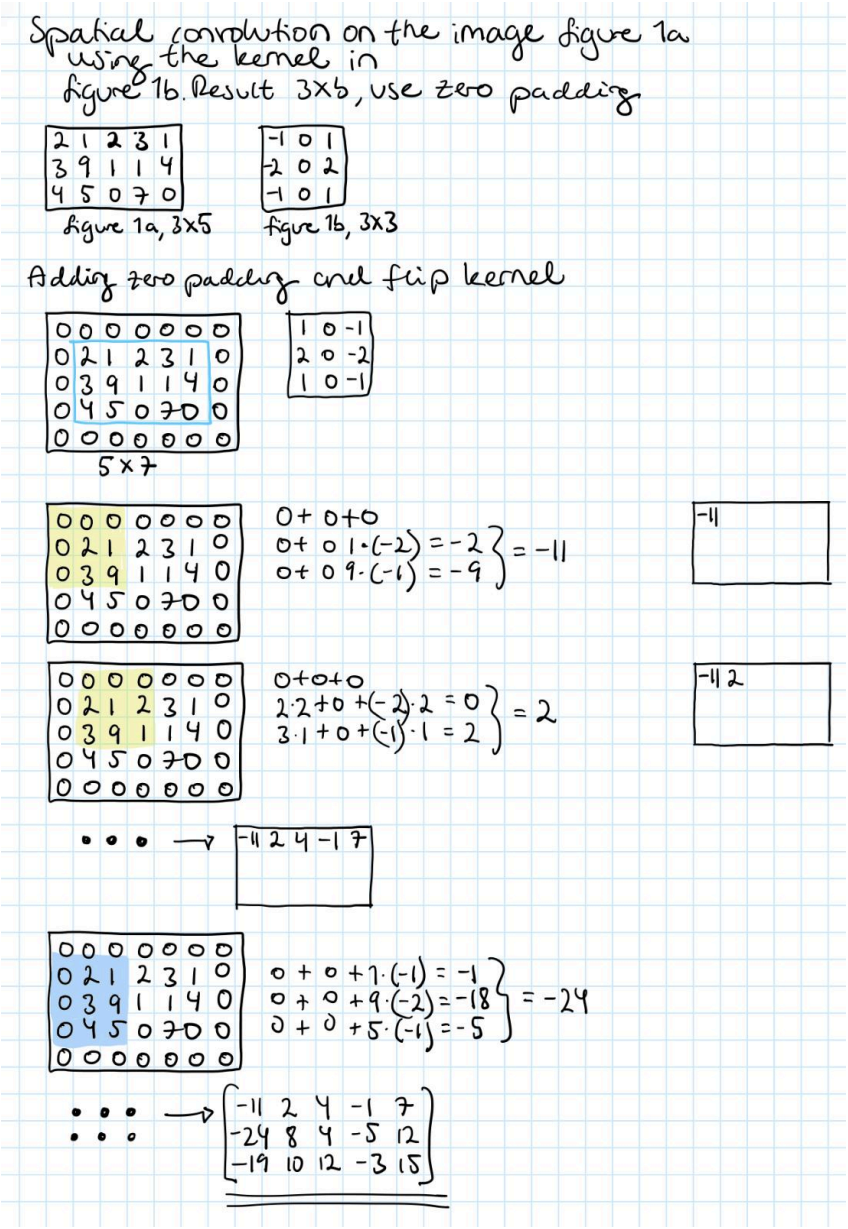
Report assignment group 114

Olof Ljunggren and Sara Sveen

Task 1a



Task 1b

- (iii) Max Pooling introduces some translational invariance. Since it decreases the spatial resolution.

Task 1c

- Given that the kernel size is smaller than the input image: the answer should be 3. The number of filters does not matter. The answer can be calculated as:

  $w_{new} = \frac{w-f+2p}{s} + 1 = /s = 1, f = 7/ = w + 2p - 6 \Rightarrow$ **p = 3**

Task 1d

- Using the same equation as in 1c and only consider one color channel:

  s=1, p=0, f=?, w = 512, w_{new}=508 $\Rightarrow 508 = \frac{512-f+0}{1} + 1$, f=5. Also symmetry gives a symmetric kernel: **Kernel size (5 x 5)**

Task 1e

- We have an even sized image and a 2x2 kernel with stride 2. This means that every pair of pixels will be merged to 1, both horizontally and vertically. So the new image size is 508/2 = 254. This can also be derived using the formula once again. Also here there is symmetry so: **Feature map size (254 x 254)**

Task 1f

- Same formula again with s=1, p=0, f=3, w = 254

  $\Rightarrow \quad w_{new} = \frac{254-3+0}{1} + 1 = 252.$

  So the new **feature map size (252 x 252)**

Task 1g

First, we want to decide the output size for every layer given s=1, p=2, f =5,w=32:

Layer 1: $\Rightarrow \quad w_{new} = \frac{32-5+4}{1} + 1 = 32$

The convolving layers do not change the image size.

Every maxpooling will be dividing the width with 2. This means that the layer 3 output will have a size of 32/2^3 = 4.

Now we can count the number of weights:

Layer 1: Kernels: 32*3*5*5, Bias: 32, in total 2 432

Layer 2: Kernels: 64*32*5*5, Bias: 64, in total 51 264

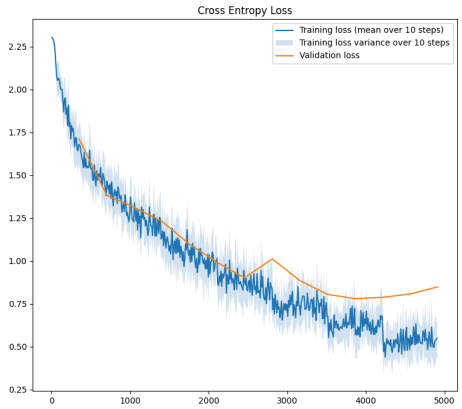
Layer 3: Kernels: 128*64*5*5, Bias: 128, in total 204 928

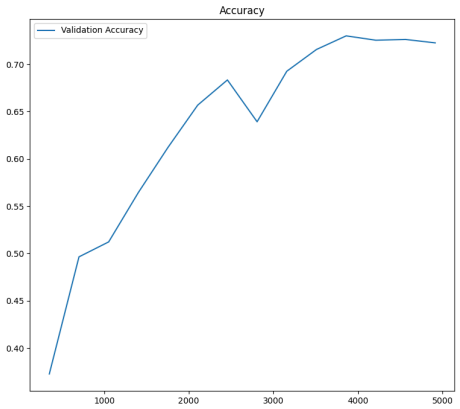
Layer 4: Fully connected: Image size * Image size * Num Images * Output layers + Bias = 4 * 4 * 128 * 64 + 64 = 131 136

Layer 5: Fully connected: 64 * 10 + 10 = 650

The total number of trainable parameters is: <u>390 410</u> parameters

Task 2a





Task 2b

Calculating forward propagation and using the results when the training has stopped gives:

Final training accuracy = 0.8445

Final validation= 0.7226

Final test accuracy = 0.72

Calculate this over the entire train/val/test datasets.

Task 3a

Our CNN model is architected for the CIFAR-10 dataset, comprising color images with 3 channels. The primary objective is to classify images into 10 distinct classes. So the initial depth is 3 and also every convolutional layer uses size 5 kernels with stride 1, padding 5. It looks like this:

nn.Conv2d(input_size, output_size, kernel_size=5, stride=1, padding=2R)

and the maxpool layers looks like:

nn.MaxPool2d(2, 2)

Model Architecture:

The model consists of the following layers:

Layer	Layer Type	Number of Hidden Units	Activation Function
1	Conv2D	32	ReLU
1	Conv2D	64	ReLU
1	MaxPool2D	-	-
1	BatchNorm2D	-	-
2	Conv2D	64	ReLU
2	MaxPool2D	-	-
2	BatchNorm2D	-	-
3	Flatten	-	-
3	Fully-Connected	128	ReLU
4	Fully-Connected	10	(Softmax)

Training Details:

- **Optimizer**: Adam optimizer.
- **Learning rate**: Learning rate $\alpha = 0.0002$.
- **Loss Function**: Cross-Entropy Loss (implicitly includes Softmax activation for the output layer).
- **Batch Size**: 64.
- **Epochs**: 10.
- **Early Stopping**: Implemented with a count of 4, based on validation loss.
- **Learning Rate Schedule**: Fixed rate throughout training without decay.
- **Weight Initialization**: Default PyTorch initialization for the layers (typically Kaiming uniform for nn. Conv2d and Xavier uniform for linear layers).
- **Regularization**: Batch normalization is applied after each convolutional layer and the first fully connected layer to stabilize the learning process and accelerate convergence. Also L2 normalization with weight decay factor 5e-4 is used.
- **Data Augmentation:** We performed augmentation with RandomHorizontalFlip and data normalization. This was done with the torchvision transform package.

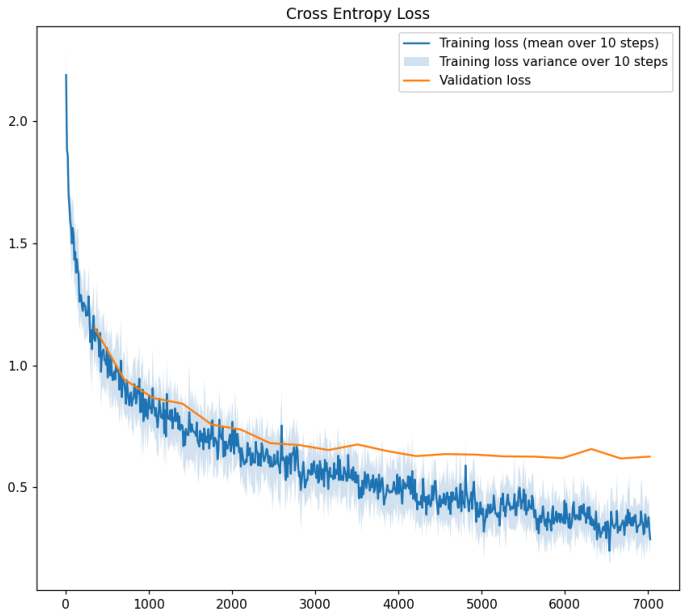
This configuration aims to optimize classification accuracy while ensuring efficient training within a quite reasonable computational budget. The model was trained using the provided Trainer class, which handles the training process, including the implementation of early stopping and recording of the training history.

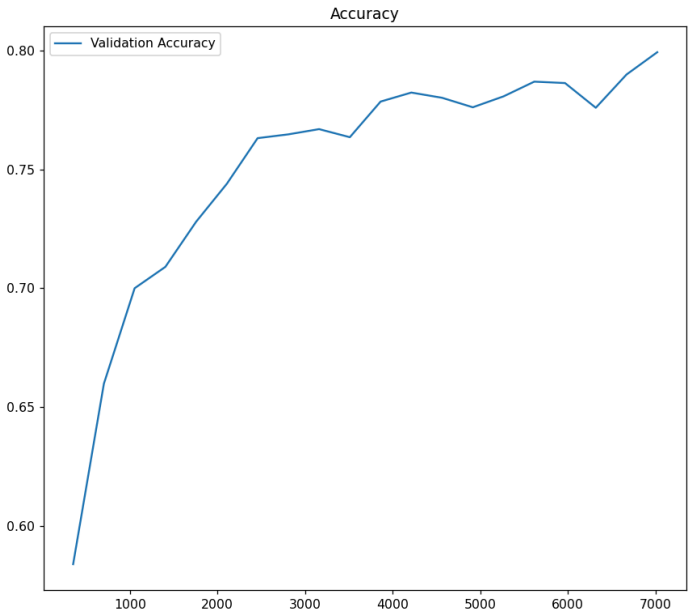
Task 3b

The performance of our final model is summarized in the following table:

Matric	Value
Final Train Loss	0.287
Train Accuracy	0.906
Validation Accuracy	0.805
Test Accuracy	0.801

Track of the training vs validation loss can be seen here:





Task 3c

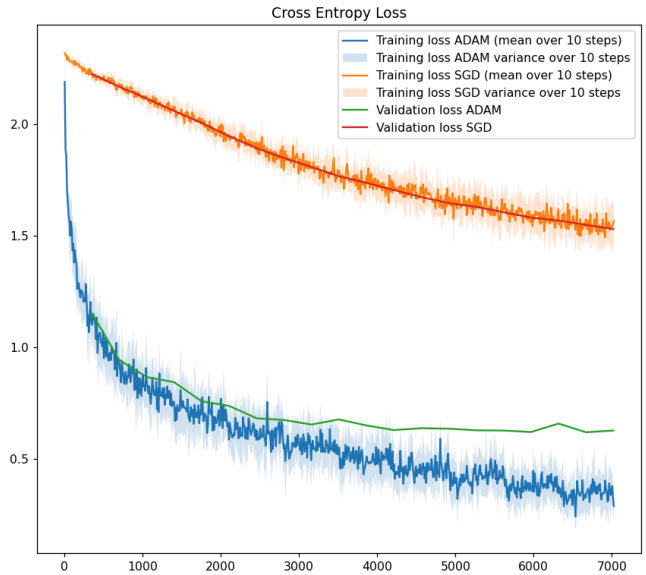
Throughout the experimentation phase, several techniques were found to be particularly beneficial:

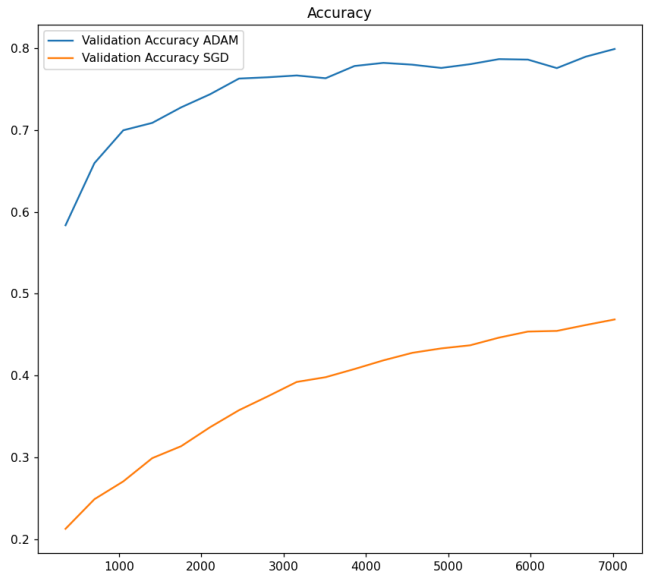
- **Normalization**: Applying batch normalization facilitated faster training and higher overall accuracy, likely due to the reduced internal covariate shift.
- **Adam Optimizer:** Switching to Adam from SGD resulted in faster convergence and required less fine-tuning of the learning rate. Adam worked fine with a low learning rate.
- **Data augmentation**
  Random horizontal flip was very good to improve the accuracy on the validation and test set. Random rotation was also tried but without the same good results.

But there were also bad results:

- **Sigmoid instead of ReLU:**
  This was really bad and it did not converge to anything good at all. This is probably because of the vanishing gradient problem.
- **Kernel Size:**
  Reducing the kernel size to 3x3 got an improvement in model training accuracy, but we lost accuracy on the test and validation set.

Task 3d

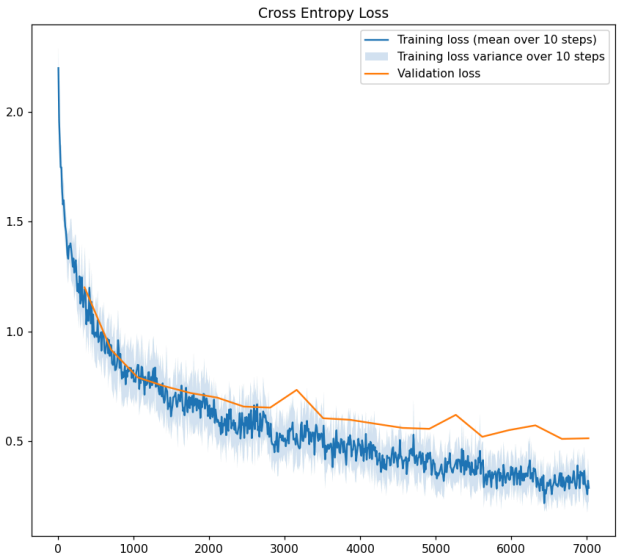


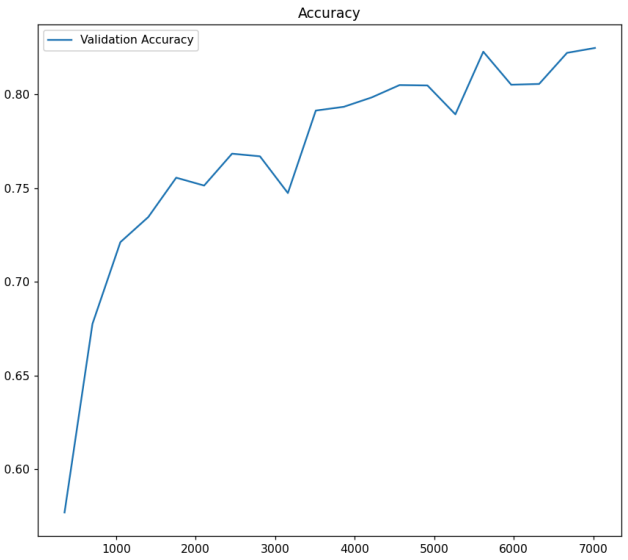


This is where the model improved the most for this learning rate. After fine tuning the learning rate for that model maybe it would be the augmentation, batch normalization or extra layer that made the best progress.

Task 3e

After improvement with an extra convolution step we get the following:





Final test accuracy = 0.8165 = 81%+

Task 3f

Examining the performance of our best model, which utilizes kernel sizes of 5 with batch normalization, we observe from the loss and accuracy plots:

- **Validation Loss:** The validation loss decreases over time, which suggests that the model is learning and generalizing well to the validation set. There is no evidence of an increase in validation loss, which is commonly associated with overfitting.
- **Validation Accuracy:** The validation accuracy increases steadily and plateaus around 0.8. This leveling off, rather than peaking and declining, indicates that the model has not overfit to the training data. An overfit model

would likely exhibit a high training accuracy with a decrease in validation accuracy over time. This is probably handled by the early stop.

- **Training Loss:** The training loss curve is also consistently decreasing and has not reached a plateau, suggesting that the model could potentially improve further with more training epochs. This might indicate that the model is not yet overfitting, as it continues to learn from the training data. But this extra improvement for the train set might also lead to overtraining and decrease in accuracy for the validation and test.
- **Gap between Training and Validation Metrics:** The gap between the training loss and the validation loss is not significantly widening, and the validation accuracy mirrors the trend in training accuracy. This further supports the conclusion that the model is not overfitting.

From these observations, we can infer that our model is neither overfitting nor underfitting significantly. The training process seems balanced, with the model learning generalized features that perform well on unseen data. However, the accuracy plateaus could be a sign of the model reaching its capacity with the current architecture and training regimen. It would be advisable to experiment with further architectural changes, regularization techniques, or training for more epochs to see if the performance can be enhanced without overfitting. So, the tendencies for overfitting are not too obvious but it might be happening since the final loss gradient seems better for the training than the validation data.

Task 4a

Final test accuracy = 0.9014