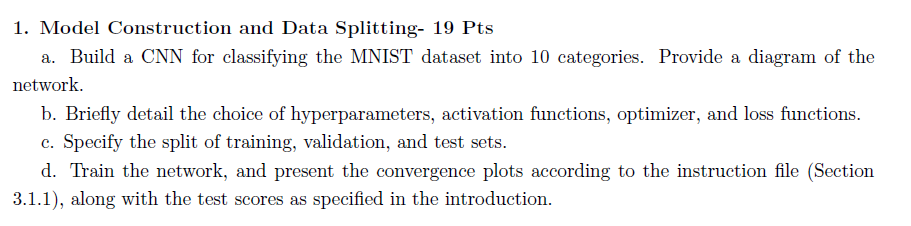
Home Work Assignment 1

Alon Zeltser - 029644119

Nadav Amir - 308339860

**Question 1**



**Answer**

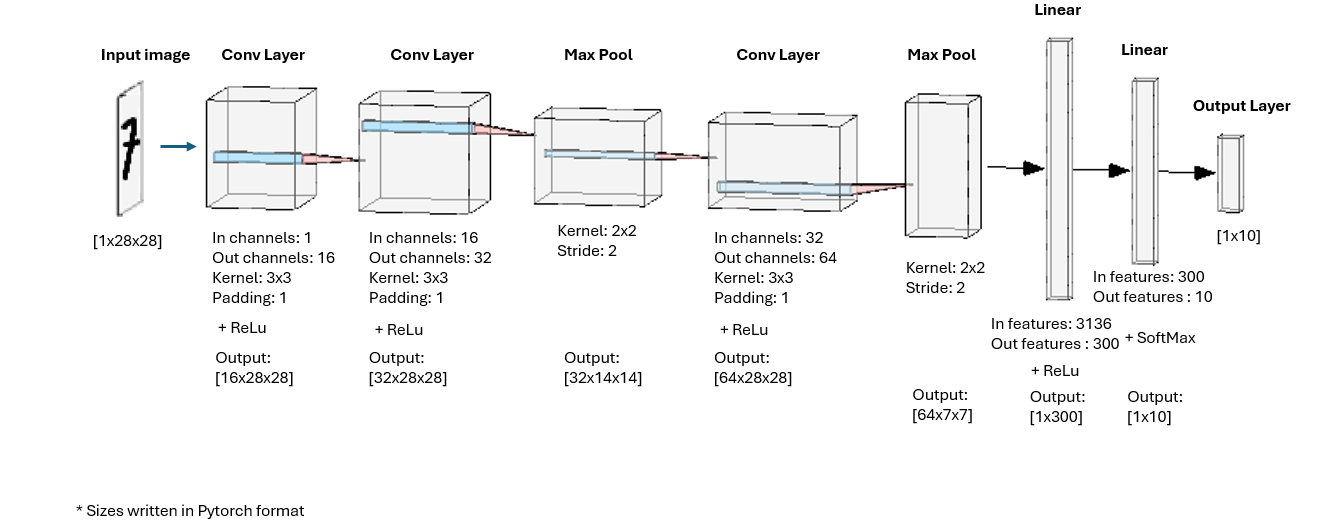
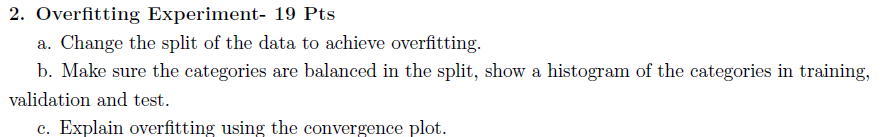


Figure 1 – Network architecture

1. We followed a common network architecture that is used for classification problems, starting with convolutional layers and continuing with reduction to fully connected layers. ReLU was used as the activation function to introduce non-linearity and minimize vanishing gradients phenomena, while SoftMax was applied at the output layer for probabilistic multi-class predictions[[1]](#footnote-1). A mini-batch size of 264 provided a balance between computational efficiency and convergence stability. We chose a learning rate of 0.001 for steady training and ran for 20 epochs to ensure sufficient learning without overfitting. Adam Optimizer was selected for its adaptive capabilities (such as per parameter learning rate and momentum). Cross-entropy loss was used, as it is suitable for multi-class classification problems.
2. We split the Train dataset into 80%/20% split resulting overall - Train: 48,000 images, Validation: 12,000 images, Test: 10,000 images (Given as a separate dataset).
3. See the next page:

|  |  |
| --- | --- |
| Figure 2 – Test Set F1 Score for each category + Macro (Average) Value | Figure 3 – Test Set Confusion Matrix |
| Figure 4 – Training vs. Validation Loss Progress Graph | |

**Question 2**



**Answer**

1. We split the Train dataset into 99.8%/0.2% split resulting overall - Train: 120 images, Validation: 59880 images, Test: 10,000 images (Given as a separate dataset).
2. We used *StratifiedShuddleSplit* to make sure that the data is balanced:

|  |  |  |
| --- | --- | --- |
| Figure 5 – Train Set Class Distribution | Figure 6 – Validation Set Class Distribution | Figure 7 – Test Set Class Distribution |

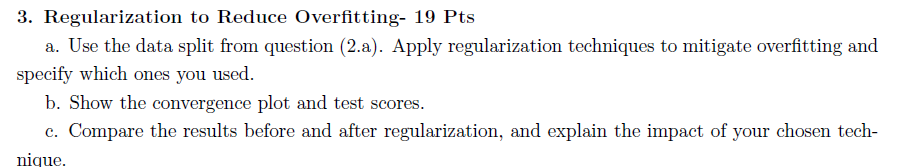
1. According to Fig 8, around epoch 13 the validation loss detaches from the training loss and begins to rise slightly, while the training loss continues converging toward zero. This suggests the model is starting to "memorize" the training set rather than generalizing, or in other words overfit.

A graph of a train vs. vallation loss

AI-generated content may be incorrect.

Figure 8 – Training vs. Validation Loss Progress Graph – Over Fitting Experiment

**Question 3**



**Answer**

1. We Applied three regularization techniques separately:
   1. Early stopping with patience of 2 epochs
   2. Dropout – we applied dropout with probability 0.05 after each layer in the model (can see specification in the attached code)
   3. L2 regularization
2. Graphs are attached for each method:

**Early stopping with patience of 2 epochs**

|  |  |
| --- | --- |
| Figure 9 – Test Set F1 Score for each category + Macro (Average) Value – Over Fitting + Early Stopping | Figure 10 – Test Set Confusion Matrix – Over Fitting + Early Stopping |
| Figure 11 – Training vs. Validation Loss Progress Graph – Overfitting + Early Stopping | |

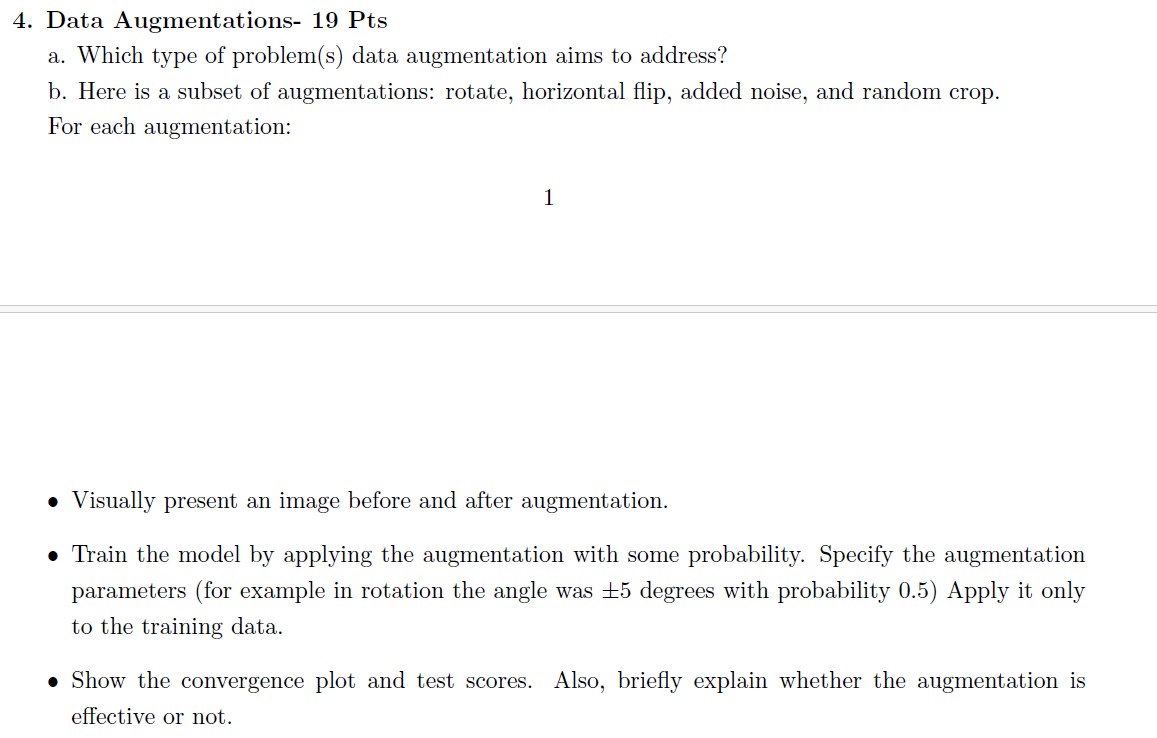
**Dropout**

|  |  |
| --- | --- |
| Figure 12 – Test Set F1 Score for each category + Macro (Average) Value – Over Fitting + Dropout | Figure 13 – Test Set Confusion Matrix – Over Fitting + Dropout |
| Figure 14 – Training vs. Validation Loss Progress Graph– Over Fitting + Dropout  **L2 Regularization** | |
| Figure 15 – Test Set F1 Score for each category + Macro (Average) Value – Over Fitting + l2 regularization | Figure 16 – Test Set Confusion Matrix – Over Fitting + l2 regularization |
| Figure 17 – Training vs. Validation Loss Progress Graph – Over Fitting + l2 regularization | |

1. Comparison. Attached is a table with macro f1 score for each regularization scenarios and for the non-regularized one.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **Without regularization** | **Early stooping** | **Dropout** | **L2** |
| **Macro F1** | 0.7645 | 0.7701 | 0.8023 | 0.7865 |

Overall, the regularization techniques produced mixed results, with some training sessions showing only slight improvements over the non-regularized model. The following results are from sessions that outperformed the non-regularized model. Early Stopping: This method stops training when the validation performance stops improving. Dropout: Randomly deactivating a small subset of neurons during training, forcing the network to learn more robust features. We examined several dropout configurations, and the selected setup-applying a small dropout after each layer, appeared to provide adequate results. L2 Regularization: Adding a penalty for large weights to the loss function, L2 regularization encourages simpler models that generalize better.



A screenshot of a computer

AI-generated content may be incorrect.

**Answer**

1. Augmentation aims to address several problems when training a model
   1. **Bias Reduction**: When the data set is, for example 1) too small, or 2) unbalanced, Augmentation can be used to expand it, adding more trainable data to the network.
   2. **Generalization and Prevention of Overfitting**: Augmentation makes models more robust and reliable by mimicking real world variations (shifts, flips, rotations, etc). This approach reduces the risk of overfitting, as the model is less likely to memorize specific patterns from the training data. For example, if all training images are captured during the day, augmentation can help the model adapt to various lighting conditions that may be encountered in a real-world environment.
2. Visual representation (left to right), as shown in Figure 18:

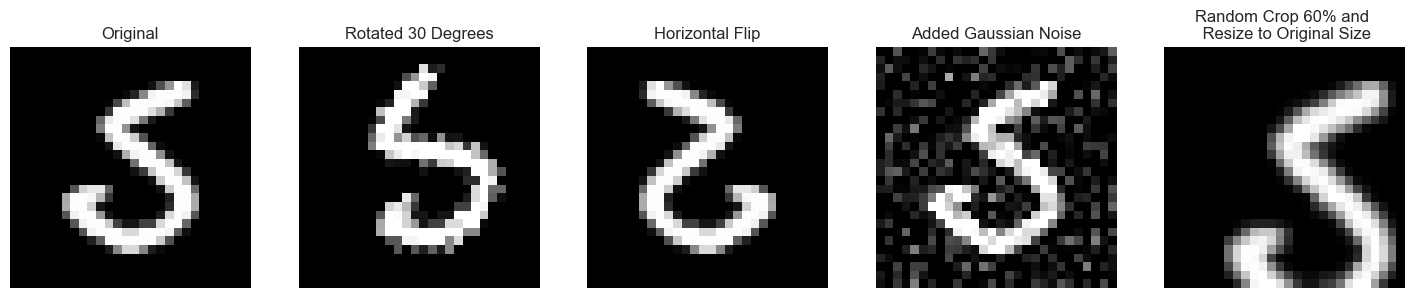


Figure 18 – Visual Representation of Augmentations

1. We ran the augmentation on Section 2 split. We applied separately:
   1. Rotation: -20 degrees to +20 degrees with probability of 0.20.
   2. Horizontal flip with probability of 0.20.
   3. Gaussian noise with amplitude of 0.2 with probability of 0.20.
   4. Random crop of between 70 to 100% of the image with probability of 0.20.
2. Graphs + explanations are attached:

**Rotation**

Rotation Might be helpful if the rotation angle is small. Adding variation in handwriting orientation may contribute to generalization without heavily distorting digits. In our example, this augmentation did not improve results.

|  |  |
| --- | --- |
| Figure 19 – Test Set F1 Score for each category + Macro (Average) Value – Over Fitting + Rotation | Figure 20 – Test Set Confusion Matrix – Over Fitting + Rotation |
| Figure 21 – Training vs. Validation Loss Progress Graph – Over Fitting + Rotation | |

**Horizontal Flip**

Not suitable for MNIST. Flip changes digit meanings (e.g., '2' or '5'), confusing the model (might be suitable for symmetric digits only). Performance got worse:

|  |  |
| --- | --- |
|  |  |
| |  |  | | --- | --- | | Figure 21 – Test Set F1 Score for each category + Macro (Average) Value – Over Fitting + Horizontal flip | Figure 22 – Test Set Confusion Matrix – Over Fitting + Horizontal flip | | Figure 23 – Training vs. Validation Loss Progress Graph – Over Fitting + Horizontal flip | | | |

**Random Crop**

Random crop might increase spatial robustness if crops are small enough. Large crops can cut off parts of digits, making them hard to recognize. It can be seen that our choice of crop (relative small) improved the model.

|  |  |
| --- | --- |
| Figure 24 – Test Set F1 Score for each category + Macro (Average) Value – Over Fitting + Random Crop | Figure 24 – Test Set F1 Score for each category + Macro (Average) Value – Over Fitting + Random Crop |
| Figure 25 – Test Set F1 Score for each category + Macro (Average) Value – Over Fitting + Random Crop | |

**Noise**

Added noise could be helpful if the test data includes camera noise or distortions, but we are not sure this is the case here. This augmentation reduced test performance:

|  |  |
| --- | --- |
| Figure 26 – Test Set F1 Score for each category + Macro (Average) Value – Over Fitting + Noise | Figure 27 – Test Set F1 Score for each category + Macro (Average) Value – Over Fitting + Noise |
| Figure 28 – Test Set F1 Score for each category + Macro (Average) Value – Over Fitting + Noise | |

1. Although softmax is implied for multi-class classification, we did not apply it explicitly, as PyTorch’s CrossEntropyLoss includes it internally. [↑](#footnote-ref-1)