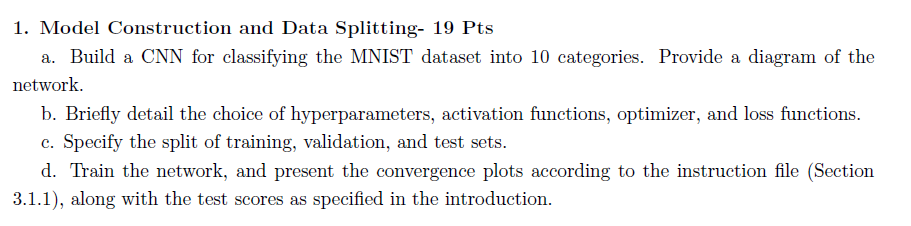
Home Work Assignment 1

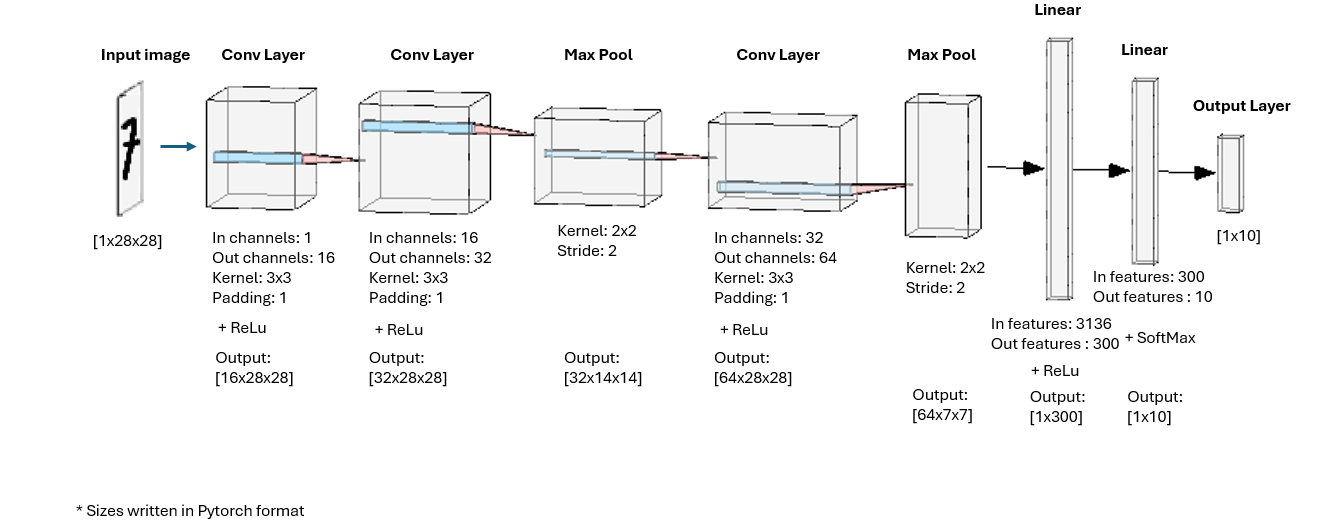
Alon Zeltser -

Nadav Amir - 308339860

**Question 1**



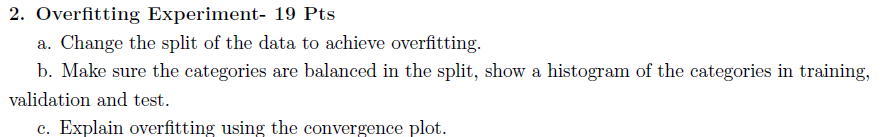
**Answer**



1. We followed a basic CNN structure. 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 learning capability. Cross-entropy loss was used, as it is suitable for multi-class classification problems.
2. Train: 48,000 images (68.57%), Validation: 12,000 images (17.14%), Test: 10,000 images (14.28%)
3. See the next page:

|  |  |
| --- | --- |
| Figure 3 – Test Set F1 Score for each category + Macro (Average) Value | Figure 2 – Test Set Confusion Matrix |
| Figure 1 – Training vs. Validation Loss Progress Graph | |

**Question 2**



**Answer**

1. We changed the split to be - Train 120 images (0.17 %), Validation 59880 images (85.54%), Test 10,000 images (14.28%)
2. We used *StratifiedShuddleSplit* to make sure that the data is balanced:

|  |  |  |
| --- | --- | --- |
| Figure 4 – Train Set Class Distribution | Figure 5 – Validation Set Class Distribution | Figure 6 – Test Set Class Distribution |

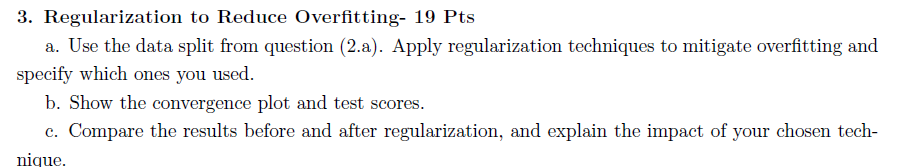
1. According to Fig 7, around epoch 12 the validation loss detaches from the training loss and begins to rise slightly, while the training loss continues converging toward zero. This suggests the model is starting to "memorize" the training set rather than generalizing.

A graph of a train vs. vallation loss

AI-generated content may be incorrect.

Figure 7 – 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 8 – Test Set F1 Score for each category + Macro (Average) Value – Over Fitting + Early Stopping | Figure 9 – Test Set Confusion Matrix – Over Fitting + Early Stopping |
| Figure 10 – Training vs. Validation Loss Progress Graph – OverFitting + Early Stopping | |

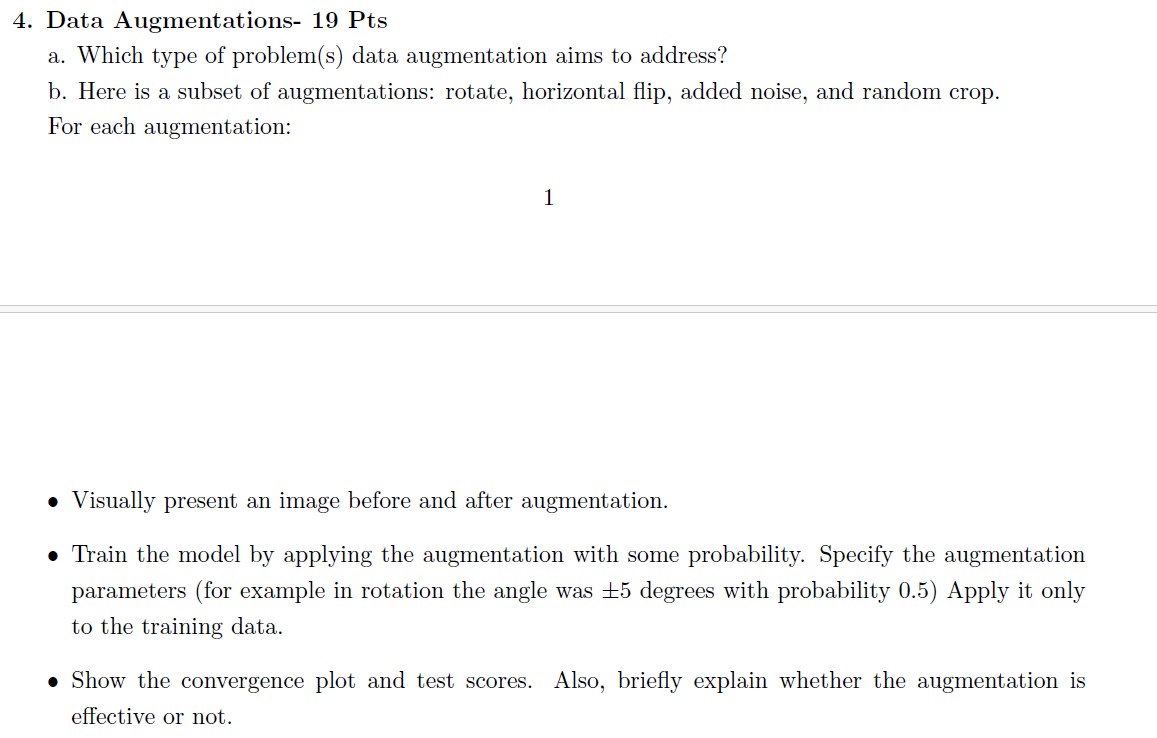
**Dropout**

|  |  |
| --- | --- |
| Figure 11 – Test Set F1 Score for each category + Macro (Average) Value – Over Fitting + Dropout | Figure 12 – Test Set Confusion Matrix – Over Fitting + Dropout |
| Figure 13 – Test Set F1 Score for Each Category + Macro (Average) Value – Over Fitting + Dropout  **L2 Regularization** | |
| Figure 14 – Test Set F1 Score for each category + Macro (Average) Value – Over Fitting + l2 regularization | Figure 15 – Test Set Confusion Matrix – Over Fitting + l2 regularization |
| Figure 16 – Test Set F1 Score for each category + Macro (Average) Value – 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. **Limited Training Data**: When the data set is too small, Augmentation can be used to expand it with more training examples.
   2. **Generalization and Prevention of Overfitting**: By simulating real-world variations, augmentation makes models more robust and reliable. This approach reduces the risk of overfitting, as the model is less likely to learn noise or overly 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.
   3. **Bias Reduction**: Augmentations, similar to the use of synthetic data, can be used to artificially increase the presentation of under presented classes or conditions
2. Visual representation (left to right), as shown in Figure 17:

Original, rotation of 30 degrees counterclockwise, horizontal flip, Added Gaussian noise scaled by 0.2 and clamping between 0 and 1 to prevent exceeding valid limits, Random crop of 60% and resize to original image size.

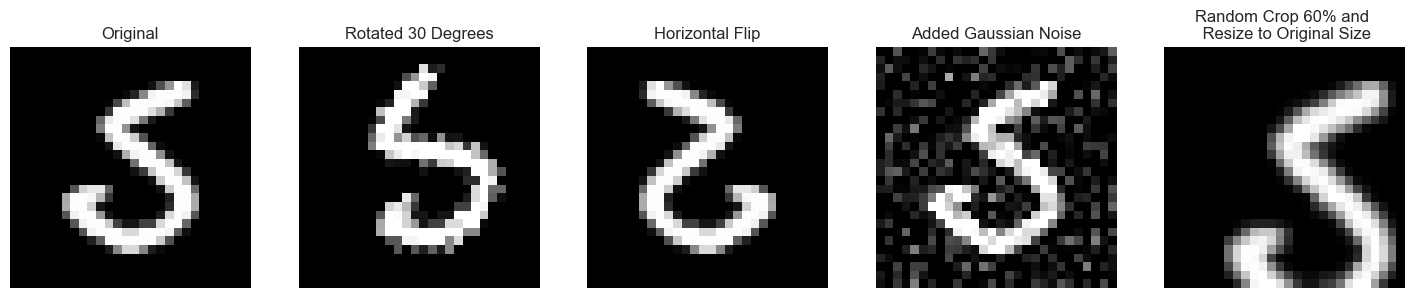


Figure 17 – Visual Representation of Augmentations

1. We ran multiple training sessions and observed that test performance generally worsened as the augmentation probability increased. Even moderate probabilities had little impact on the model since the dataset very big. Below, we show the effect when most of the images are augmented. The parameters used were:
   1. Rotation: -20 degrees to +20 degrees with probability of 0.9
   2. Horizontal flip with probability of 0.9
   3. Gaussian noise with amplitude of 0.2 with probability of 0.9
   4. Random crop of between 50 to 100% of the image with probability of 0.9
2. For each augmentation method, we briefly write whether it is beneficial for MNIST:
   1. **Rotation** - might be helpful if the rotation angle is small. Adding variation in handwriting orientation may contribute to generalization without heavily distorting digits.
   2. **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)
   3. **Gaussian noise** - Likely not helpful. It could be helpful if the test data includes camera noise or distortions, but we are not sure this is the case here.
   4. **Random** **Crop** – May increase spatial robustness if crops are small enough. Large crops can cut off parts of digits, making them hard to recognize.

|  |  |
| --- | --- |
| Figure 17 – Test Set F1 Score for each category + Macro (Average) Value – Augmentation | Figure 18 – Test Set Confusion Matrix – Augmentation |
| Figure 19 – Test Set F1 Score for each category + Macro (Average) Value – Augmentation | |

We can see that the performance on asymmetrical digits worsens. We observed that this effect becomes significantly more pronounced as the horizontal flip augmentation probability approaches 1.

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)