Multimodal Deep Learning on MNIST data

Vineet Gandham
Computer Science Department
Texas A&M University
College Station, TX 77845
vineet.gandham@tamu.edu

Abstract

This project focuses on predicting the handwritten digits shown on an image and the corresponding audio of the digit. The objective is to develop a model capable of accurately recognizing digits from both image and audio inputs. The model is trained on MNIST dataset containing roughly 60000 images and audio recordings. To achieve this, a convolutional neural networks (CNN) fusion model with long short-term memory(LSTM) layers was implemented. This hybrid architecture integrates CNNs for image processing with LSTM networks for sequential data analysis. The model achieved a remarkable test accuracy of 0.99. This project leveraged multiple techniques to generate embedding of a common task, which shows the potential and importance of multi-modal approach.

1 Introduction

This project addresses the problem of multimodal classification, where the goal is to integrate image and audio data to improve the accuracy and robustness of classification models. In this project, to predict the single digit given as handwritten image and a audio recording. This approach is fruitful in tasks such as multimedia content analysis where both visual and auditory cues are crucial for comprehensive understanding. To address this problem, I designed a multimodal neural network architecture comprising two encoders: one for images and one for audio. The Image Encoder uses convolutional neural layers to process visual information, extracting spatial features. The Audio Encoder uses an LSTM network to capture temporal dependencies and dynamics in the audio data. Outputs from both encoders are then fused, allowing the model to learn from the combined feature space effectively. By the final training epoch, the model achieved a Validation Accuracy of 99.42% and a Validation F1 Score of 99.42%, indicating high precision and recall.

2 Methodology

2.1 Data Preprocessing

The dataset is a multimodal MNIST data. The MNIST dataset is a collection of 60000 28x28 pixel gray-scale images of handwritten digits ranging from 0 to 9. Each image is labeled with the corresponding digit it represents. Additionally there is audio of length 507 for each data sample. Fig 1 shows the data distribution of the dataset. We have observed that there is roughly the same amount of samples for each label. Fig 2 shows a sample of the image data.

There are multiple ways to pre-process and augment image data. Random rotations, color jitters, cropping, horizontal and vertical flips are some standard practises. For this project, the best results were when the image data was normalized with mean 0.1 and standard deviation of 0.3. I've also augmented by randomly flipping the data horizontally with probability of 0.5.

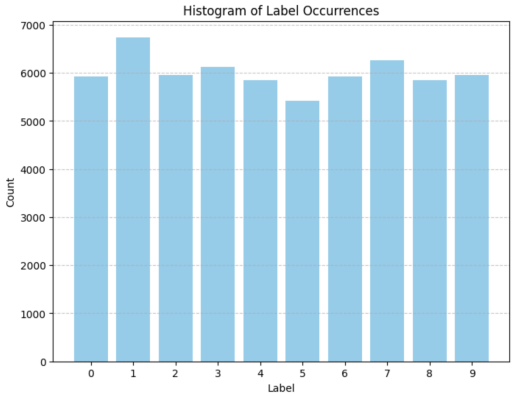

Figure 1: The distribution of labels

Figure 2: Sample images of MNIST dataset

Since, audio recognition is a sequence modelling problem, I've divided each instance of audio into 10 time steps of length 50 each. To match the dimensions, I omitted the last 7 sequences. This pre-processing step is essential to get meaningful sequences for RNNs like LSTMs.

For the purposes of training and evaluating, a random validation set of 20 percent of the 60000 examples were used.

2.2 Model Design

The architecture consists of three main components: an image encoder, an audio encoder, and a classifier.

The image encoder is a convolutional neural network (CNN) that extracts features from the input image. It has two convolutional layers followed by batch normalization, ReLU activation, max pooling, and dropout layers. These layers help the network learn hierarchical representations of the image data, reducing the dimensions while increasing the depth of features. The extracted image features are then passed through 3 fully connected layers with batch normalization, ReLU activation, and dropout for dimensionality reduction.

On the other hand, the audio encoder is implemented using a long short-term memory (LSTM) network, which is capable of capturing temporal dependencies in sequential data like audio. The LSTM layers process the input audio sequence and produce a fixed-size representation of the audio features. I've used a stacked LSTM of 2 layers, with output dimension of 36 features. The hidden state is extracted and a skip connection from both layers are concatenated along the feature dimension. This concatenation increased the richness of information available to the model.

Finally, the output of the image encoder and the audio encoder are **added** and fed into a series of fully connected layers, ending in a softmax layer to produce the final class predictions. The model is trained end-to-end using backpropagation to minimize the classification loss(Cross Entropy Loss).

As part of hyperparameter tuning, I've experimented with different types of pooling layers, kernel sizes, dropout, batch normalization, number of layers, unidirectional and bidirectional LSTMs. On the other hand, the parameters that weren't tuned during training was initialization of trainable weights and how to fuse the embeddings.

Table 1 outlines the validation F1 score as a result of different types of fusion methods. As we can observe, different fusion methods have a marginal difference in validation F1 score, while keeping rest of the architecture majorly the same.

Table 1: Comparison of Fusion Methods with F1 Scores

Fusion Method	F1 Score
Concatenation	0.993
Addition	0.994

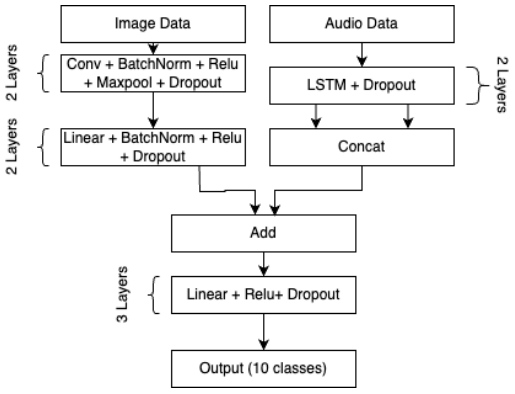

Figure 3: Diagram of multi model network

Figure 3 is a diagram of the best performing architecture. Both image and audio data is fused and then linear layers are implemented on top of them. Figure 4 contains the same model with more elaborate details.

2.3 Model Training

In this study, I trained my model using a step-by-step approach to ensure effectiveness and reliability. First I set up the model components and moved them to the GPU(A100) for training. I chose Adam optimizer with a learning rate of 0.001 and Cross Entropy Loss as the training criteria. Splitting the dataset into 80% for training and 20% for validation, I used DataLoader to manage the data efficiently in batches with a batch size of 32. Setting the epochs between 10 to 20 balanced overfitting and time constraints, as each epoch took 20 minutes on the CPU and 2 minutes on the GPU. Throughout the training process, I constantly evaluated the model's performance on the validation set to make sure it was learning effectively. This approach ensured that my results are trustworthy and can be easily replicated for further studies.

The objective of this task is to see which of the 10 classes does a sample belong to. Therefore, I kept the final output to be of 10 dimensions and fed it into Pytorch's CrossEntropyLoss function.

```
MultimodalClassifier(
  (image_encoder): ImageEncoder(
    (cnn): Sequential(
      (0): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (4): Dropout2d(p=0.25, inplace=False)
      (5): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (6): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (7): ReLU()
      (8): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (9): Dropout2d(p=0.25, inplace=False)
    )
    (fc): Sequential(
      (0): Linear(in_features=1568, out_features=100, bias=True)
      (1): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): Dropout(p=0.5, inplace=False)
      (4): Linear(in_features=100, out_features=72, bias=True)
      (5): BatchNorm1d(72, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (6): ReLU()
      (7): Dropout(p=0.5, inplace=False)
    )
  )
  (audio_encoder): AudioEncoder(
    (lstm): LSTM(50, 36, num_layers=2, batch_first=True, dropout=0.3)
  )
  (fc): Linear(in_features=72, out_features=60, bias=True)
  (fc2): Linear(in_features=60, out_features=46, bias=True)
  (fc3): Linear(in_features=46, out_features=10, bias=True)
)
```

Figure 4: Detailed architecture of multi model network

2.4 Hyperparameter Tuning

I made several iterative adjustments to enhance model performance. Initially, I experimented with different structures such as multiple convolution, linear, and LSTM layers in both the image and audio encoders to improve feature learning by increasing depth. Then I added dropout with batch normalization to stabilize and speed up the convergence by normalizing the inputs to each layer. As the epochs progressed, I realised that there was a lot of noise from the linear layers as the model was unable able to get relevant features so I adjusted the architecture of the fully connected layers, such as altering the number of neurons and layers. This lead to enhanced learning capabilities. The validation accuracy and F1 scores improved with each adjustment, indicating a more robust model capable of better handling the intricacies of the input data. This systematic approach of experimenting with architectural changes and observing their impact on model validation metrics was crucial in effectively tuning the hyper-parameters.

3 Results

I experimented with multiple models, but I will summarize 3 where the learnings from first 2 models led to the most accurate model - Model C.

3.1 Model A

The image encoder comprises **two convolutional layers with 16 and 32 filters**, respectively, followed by max-pooling operations, ultimately producing a feature vector of dimension 128. I did not add dropout nor any linear layers after convolutions. Conversely, the audio encoder employs an **LSTM network with a hidden size of 32**, extracting temporal dependencies from the input audio sequences to generate a 32-dimensional feature representation. Fusion of the modalities occurs through concatenation, combining the 128-dimensional image features with the 32-dimensional audio features, resulting in a combined feature vector of dimension 160. This concatenated representation is then fed into a **single fully connected** layer to produce class predictions. **This model resulted in validation F1 score of 0.986 and test F1 score of 0.988.**

3.2 Model B

In Model B, I introduced batch normalization layers after both convolutional and **2 linear layers** in the image encoder and audio encoder, aiming to learn features and accelerate the training process with a little bit of regularization effect. Additionally, dropout layers with probability of 0.3 were applied to the convolutional layers in the image encoder and to the LSTM layers in the audio encoder, which enhanced the model's ability to generalize by reducing overfitting. The fully connected layers in the image encoder and multimodal classifier were also augmented with batch normalization and dropout(0.5) for regularization. Furthermore, the architecture of the multimodal classifier in Model B features an additional fully connected layer with 86 neurons, followed by dropout and another fully connected layer with 54 neurons before the final output layer for class prediction.

Model B had improved regularization and more deep neural layers compared to Model A which resulted in validation **F1 score of 0.994 and test F1 score of 0.993.**

3.3 Model C

The architecture of the image encoder and multimodal classifier remained similar to Model B, with the exception of the fully connected layers in the multimodal classifier. In Model C, the number of neurons in the final fully connected layer is reduced to 46, to control model complexity and enhance generalization performance. Additionally, the fusion method is changed from concatenation to element-wise addition of image and audio features, to promote complementary interactions between modalities.

I've also included skip connections in LSTM layers where information from earlier layers is combined with the output of later layers. This enhanced the **test F1 score to 0.994, but validation F1 score was at 0.992**. My analysis is that the skip connections helped enrich spatial features while reducing the number of neurons per layer decreased overfitting.

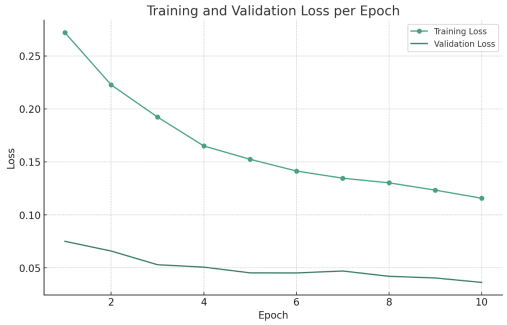

Figure 5: Validation and Train Loss Line graph

3.4 Summary

Table 2 summarises the F1 scores of each model. Figure 5 shows that the line chart displaying the training and validation loss across 10 epochs. We can observe how both metrics generally decrease over time, indicating the model is learning and improving its performance with each epoch.

Model	Validation F1 Score	Test F1 Score
A	0.986	0.988
B	0.994	0.993
C	0.992	0.994

Table 2: Validation and Test F1 Scores for Models A, B, and C

4 Conclusion

The models performed well overall, with Model C exhibiting the highest test F1 scores. This success is due to the incorporation of batch normalization and dropout regularization, which helped stabilize training and mitigate overfitting. Additionally, the fusion method of concatenating image and audio features in Model C might have helped in the better integration of multimodal information. To further improve results, fine-tuning hyperparameters such as learning rate and dropout probabilities, as well as exploring more sophisticated fusion strategies and incorporating data augmentation techniques, could enhance model performance and robustness. Experimenting with more advanced architectures, such as attention mechanisms or transformer-based models, could potentially capture more complex relationships within the data and further boost performance. Additionally utilizing CNN on converted data types such as audio spectogram or conversely, using RNN based architecture across the images could help get more deeper and meaningful embeddings.