CSCE 633 HW 5 - CNN Model for Multimodal Data

Arunachalam Venkatachalam
Department of Mechanical Engineering
Texas A&M University
College Station, TX 77840
`arun23venkat@tamu.edu`

Abstract

Handwritten digit recognition is a crucial task in computer vision with a plethora of applications. This project aims to develop a robust deep learning model fusing image and audio modalities. Two separate CNNs were trained for image and audio data, achieving 99.76% accuracy and F1 score of $\sim$ 99.8 on unseen data. The combined model demonstrated superior performance, highlighting the effectiveness of multimodal fusion for improving digit classification.

1 Introduction

This paper presents a novel approach to handwritten digit recognition using a Convolutional Neural Network (CNN) trained on a multimodal dataset incorporating both visual and auditory information. The visual data is sourced from the MNIST dataset, consisting of 70,000 handwritten digits (28x28 pixels), while the audio data is extracted from the Google Speech Commands dataset. As mentioned by Khacef et al.[KRM19], the audio data is pre-processed by extracting Mel Frequency Cepstral Coefficients (MFCC) with a framing window size of 50 ms and a frame shift size of 25 ms. The speech samples, approximately 1 second long, yield 39 time slots. Each slot extracts 12 MFCC coefficients with an added energy coefficient, resulting in a 39 x 13 = 507-dimensional vector. Standardization and normalization are applied to the MFCC features. By combining these datasets, the CNN can leverage complementary information from both modalities, potentially leading to improved digit recognition performance compared to models relying solely on visual data.

Two separate encoder models for the image and audio datasets were created. Both encoders utilized CNNs to extract spatial features from the 2D images and temporal patterns from 1D audio samples. Various combinations of 2D convolutions, dropouts, relu, max pool and fully connected layers were tested for images. Whereas for the audio data, multiple headed kernels, 1D convolutions with different kernel sizes, max pool, relu and dropouts were experimented. The outputs of both encoders were concatenated and fused via multiple fully connected layers with batch normalization. Atlast the output was passed through a softmax layer for final classification. Upon training completion, embeddings from the encoder models were extracted. These embeddings represented the learned features for each modality. In the analysis, t-Distributed Stochastic Neighbor Embedding (t-SNE) was employed to reduce the dimensionality of the embeddings to 2D for visualization. Furthermore, k-means clustering with k=10 was applied to compare the clustering results of the audio and image embeddings, offering insights into the effectiveness of the fusion strategies.

Lateef et al. [LA22] conducted a study on tuning hyperparameters of a 1D CNN model to enhance Human Activity Recognition (HAR) performance. They found that standardization of data improved average accuracy from 90.8% to 91.1%. By comparing different approaches for kernel size tuning, they determined that a kernel size of 3 achieved 90.34% accuracy, while a multi-headed CNN model with varied kernel sizes reached 91.40% accuracy. Increasing epochs improved accuracy, but a batch size over 64 led to decreased accuracy. The study suggests further exploration of hyperparameters and use of evolutionary algorithms for optimization in 1D CNN research.

2 Methodoloy

2.1 Data Pre-processing

Normalization and Standardization: The multimodal MNIST dataset is preprocessed to facilitate training. Image and audio data were already normalized, ensuring pixel values are within the range [0, 1]. This normalization helps in reducing the impact of varying pixel intensity values and can lead to faster convergence during training. Normalizing the audio data to a similar scale is important for ensuring that the model can effectively learn from both modalities.

Data conversion and Reshaping: The data is then reshaped to match the expected input format of the model, audio is reshaped to (batchsize, channels, length), while images are reshaped to (batchsize, channels, height, width). Labels are loaded and converted to tensors for model compatibility. All data are in/converted to float32.

Class Imbalance rectification: Class weights are calculated to address class imbalance, ensuring effective training across all classes. Imbalance in class would lead to bias in the learnings of the model. By assigning higher weights to underrepresented classes, the model can learn to give more importance to these classes during training, leading to better performance on all classes. This step is crucial for achieving optimal performance and generalization.

Dataloader: Custom Dataloader was created to sample image and audio inputs based on the batch size, 32 in this case, along with their labels. The input tensors along with the label tensor are bundled together and fed to the model in batches. Batch processing allows the model to process multiple samples in parallel, leveraging hardware acceleration and speeding up training.

2.2 Model Design

The model consists of two main branches for processing image and audio inputs, followed by a classification head that combines the features from both modalities for the final prediction. This model architecture is chosen because it allows for independent processing of the audio and image data streams, potentially capturing separate features from each modality before combining them for final classification.

Images are complex and multi-dimensional, whereas audio is 1D and has varying pattern for every time period. For example, images have spatial features like edges, textures, and shapes, while audio has temporal features like pitch, amplitude, and frequency. By using separate CNNs, the model can specialize in extracting modality-specific features and can learn hierarchical representations specific to each modality. In addition to that, separately processing images and audio allow for greater flexibility in the model architecture. Different architectures and hyperparameters can be used for each modality, which can lead to better generalization and performance on the task.

Image Encoder

1. **Convolution Layers:** The core of the image encoder is its three convolutional layers, each progressively extracting more intricate and hierarchical features. The first layer, with a smaller kernel size (e.g., 5x5) and 32 filters, focuses on capturing low-level visual elements like edges, corners, and basic shapes. As we progress through the network (layers conv2 and conv3), the number of filters increases (64, 128) while the kernel size is made to (3x3). This allows the network to build upon the simpler features learned earlier, enabling the extraction of increasingly complex features.
2. **ReLU Avtivation:** The ReLU (Rectified Linear Unit) activation function is applied after each convolutional layer (except the last) to introduce non-linearity into the network. This is crucial because without it the model's ability to learn the complex relationships is limited. ReLU ensures the network can learn more expressive features.
3. **Max Pooling:** Max pooling layers (2x2) are employed after some convolutional layers to reduce dimensionality of the feature maps. Done after each convolutional layer. Decreases

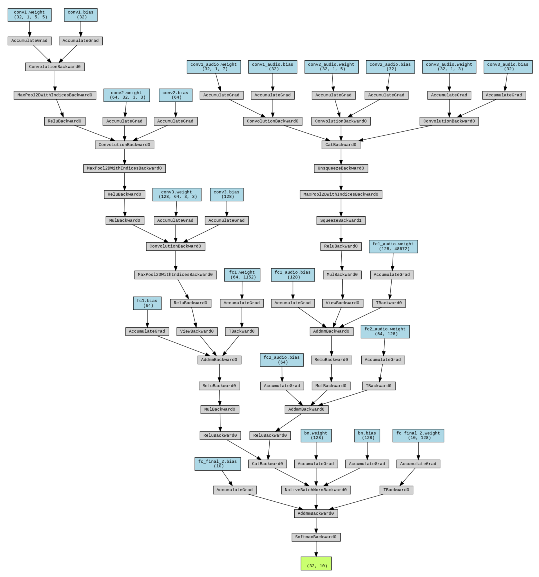

Figure 1: Graphical representation of the CNN model

the number of parameters in the network, improving computational efficiency and reducing the risk of overfitting. Second, max pooling introduces some level of translation invariance by selecting the most prominent feature within a local region. This makes the model less sensitive to small shifts in the position of objects within the image, improving robustness.

4. **Dropout:** The inclusion of a dropout layer with a probability of 0.3 helps prevent overfitting and regularize the data. Dropout layer is placed after the second conv layer and after the first fully connected layer. During training, dropout randomly sets a certain percentage of activations in the feature map to zero. This forces the network to learn features that are not reliant on any specific neuron, promoting robustness.

5. **Fully Connected Layers:** The code utilizes a fully connected layer (fc1) after flattening the feature maps from the final convolutional layer (conv3). The convolutional layers extract a large number of features organized in feature maps. Flattening these maps results in a high-dimensional vector. Fully connected layers allow the model to learn the intricate relationships between the extracted features from all convolutional layers. These relationships are crucial for discriminating between different image classes.

Audio Encoder

1. **Convolution Header:** Initial approaches to audio encoders often relied on multiple convolutional layers with varying or constant kernel sizes, achieving accuracies of up to 99.4-99.5%. However, Lateef et al. (2022) proposed a novel approach using multi-headed kernels within a single convolutional layer. This method replaces the sequential extraction of features with a parallel approach. The input data is passed through three "heads" within the same layer, each utilizing a different kernel size (7, 5, and 3) and corresponding padding (3, 2, and 1, respectively). This allows the model to capture features at various scales simultaneously. The outputs from these heads are then concatenated (output size is (96,1,507)), offering a richer feature representation compared to the traditional sequential approach. This modification, without altering the overall image encoder structure, resulted in significant accuracy improvements, reaching up to 99.7-99.8%.

2. **ReLU Activation:** The ReLU (Rectified Linear Unit) activation function is applied after each convolutional layer (except the last) to introduce non-linearity into the network.

3. **Max Pooling:** Max Pooling downsamples along the temporal dimension (for audio data). The stride determines how much the window is shifted for the next pooling operation. The values of kernel size (3), stride(1) and padding(1) were chosen post tuning the values in order to achieve the desired output format demanded by the next layer. Any other value of

stride, kernel size and padding would change the dimension from the required (32, 1, 507) to either (32, 1, 506) or (32, 1, 505) or a different one.

4. **Dropout:** The same dropout of 0.3, as in the case of the image encoder, was used to prevent overfitting. Dropouts places after the convolution layer and first fully connected layer.
5. **Fully Connected Layers:**The first fully connected layer takes the flattened output of the multi-headed kernel of the convolution layer concatenated together as input. It reduces the high-dimensional audio features to a 128-dimensional representation, enabling the model to learn more compact and abstract features. The second fully connected layer further reduces the dimensionality to 64. These fully connected layers act as feature extractors, transforming the raw audio features into a more abstract and meaningful representation.

Encoder Fusion and batch normalization

Concatenation of the image and audio data followed by batch normalization is a common approach in multimodal learning. This method allows the model to learn a more complex relationship between the two modalities compared to simple element-wise operations like addition or multiplication. Addition might lead to feature redundancy, while multiplication might overly emphasize one modality over the other. Concatenation preserves the individual characteristics of each modality, allowing the model to leverage the unique information from both sources. Batch normalization helps stabilize and speed up training by normalizing the concatenated features.

2.3 Model Training

The model was trained using batches of data containing both audio and image inputs, along with their corresponding labels. Training was conducted over multiple epochs on CPU. After each epoch, the model was evaluated on a separate validation dataset. The CrossEntropyLoss was used as loss function with modifications to incorporate class weights based on the distribution of classes in the dataset. Higher weights are assigned to minority classes. This helped improve the accuracy of the model from 99.4% to 99.5% and to learn equally from all classes thereby avoiding potential bias towards a particular class. The Adam optimizer was employed for optimizing the model's weights, providing adaptive learning rates and momentum for efficient training.

2.4 Hyperparameter Tuning

Around 9-10 different models were developed. Based on the F-1 score obtained for predictions of a particular model, new models were built progressively. All models used ReLU as activation function and 30% dropout rate.

1. **Try 1:** The initial model consisted of two convolutional layers for image encoder with kernel size 3 and padding 1, two maxpool layers with kernel size 2 and stride 2, one dropout layer and one fully connected layer. Audio encoder had 2 convolutional layers with kernel size 5 and 3 respectively with no padding, one maxpool layer with kernel size 2 and a fully connected layer. The output of both the encoders were concatenated to have 128 features which were then fed to the final fully connected layer to provide the classification. Trained for 30 epochs. F1 score - 99.4
2. **Try 2:** The second try had the same architecture, but this time the imbalance in the data was catered to. Class frequency based weights were computed and higher weights were given to minority classes. Loss function was modified to incorporate the weights. Trained for 30 epochs. F1 score - 99.5
3. **Try 3:** Altered the kernel size of first convolution layer in audio encoder from 5 to 7. Increased the number of fully connected layers from one to two in image encoder, audio encoder and post fusion layers. Batch normalized the data before feed it into the first of two fully connected layers post fusing the features from image and audio encoders. Included another dropout layer between fully connected layer 1 and layer 2 for both image and audio encoder. Added softmax layer at the very end. Trained for 50 epochs. F1 score - 98.8
4. **Try 4:** Changed kernel size of second convolution layer in audio encoder from 3 to 5. Removed the last dropout layer from image and audio encoder. F1 score - 98.5

5. **Try 5:** Increased the kernel size of max pool layer of audio encoder from 2 to 3. Removed the second fully connected layers of audio and image encoder. Trained for 30 epochs. F1 score - 99.2
6. **Try 6:** Trained model of try5 for 20 more epochs by loading the final saved model, making it a total of 50 epochs. F1 score - 99.2
7. **Try 7:** Went back to the same architecture as in Try2 but increased the number of epochs from 50 to 100. Tapped the weights at the best epoch (highest validation accuracy) and post 100 epoch completion. Predictions using the weights stored at best epoch produced a F1 score of 99.3. Predictions using the weights post training the model for 100 epochs had a F1 score of 98.9.
8. **Try 8:** Changed the kernel sizes of audio convolution layers to 3 and the kernel size of max pool layer to 2. Removed all dropout layers. Increased the number of features coming out of fully connected layers of both audio and image to 128. So, now the first conv layer post fusion will take in 256 inputs as opposed to 128 in the previous trials. Trained for 100 epochs. Tapped the weights at the best epoch (highest validation accuracy) and post 100 epoch completion. Predictions using the weights at best epoch and post the final epoch fetched the same F1 score of 99.3.
9. **Try 9:** Removed the last convolution layer, previous 2 conv layer post fusion now 1. Changed the kernel sizes of audio convolution layers to 5. Reduced the number of output features from last conv layer of audio and image back to 64. No dropout layers still. Ran the model for 100 epochs. Again, had two different weights as like the prior tries. The one at best epoch yielded a F1 score of 99.3 and the wights post 100 epoch completion provided a F1 score of 99.4.
10. **Try 10:** Completely removed the currently employed conv layers of audio and changed it to a multiheaded kernel (details mentioned under audio encoder). Introduced a third conv layer for image encoder with kernel size from first to third layer being 5,3,3. 3 max pool layers, and 2 dropout layers for audio and image encoders. Only one fully connected layer post fusion. Weights at best epoch ended up getting a F1 score of 99.6 whereas predictions using the final weights gave the best F1 score of 99.8.
11. **Try 11:** Added one more convolution layer in image encoder totalling to four conv layers with kernel sizes 7,5,3,3 resp. Added another conv layer to the audio encoder keeping the already present multi-headed kerneled conv layer with kernel size of 3 (no padding) and a max layer following it with a kernel size of 2 (no padding). 3 dropout layers in each audio and image encoders placed alternatively. Additional fully connected layer placed at the end of audio encoder. Two F1 scores again, at best epoch - 99.4 and one using the final weights - 99.5.
12. **Try 12:** Added another fully connected layer at the end of image encoder making it two for both image and audio. Reduced number of dropout layers to 2 for both image and audio. F1 score at best epoch - 99.7. F1 score post 100 epochs - 99.6.
13. **Try 13:** Changed the kernel size of conv layer to 5,5,3,3. Removed the last fully connected layers of image encoder. Removed the extra conv layer added to audio encoder in the last trial. One dropout in image encoder and two encoders in audio encoder currently. F1 score at best epoch - 99.7. F1 score post 100 epochs - 99.6.

3 Results

Try	F1 Score at Best Epoch	F1 Score post full model training
1	-	99.4
2	-	99.5
3	-	98.8
4	-	98.5
5	-	99.2
6	-	99.2
7	99.3	98.9
8	99.3	99.3
9	99.3	99.4
10	99.6	**99.8**
11	99.4	99.5
12	99.7	99.6
13	99.7	99.6

Table 1: F1 Score across different trials

Model	Accuracy
My Model	**99.76**
VGG-5 (Spinal FC)	99.15
CAMNet3	99.05
VGG8B (2x)	99.01
CN (d=32)	98.84
NSRL (log D)	98.81
ResNet-152	98.79
ResNet-14	98.75

Table 2: Accuracy comparison against SOTA

Compared to classic state-of-the-art models, my model which fuses data from two different modalities performed better. Other models in the list rely only on image data.

The image and audio embeddings are attached below. The embeddings show that the model extracted clusters associated with the digit labels.

Image embedding:
Data points belonging to the same class are more locally clustered together. All classes except for '0' are well separated and have distinct boundaries in case of image encoder. The '0' class is overlapping with most of the other classes because of its similarity in features with other classes.

Audio embedding:
In case of audio encoder, the classes are not well separated and are intermixed. But the classes are locally clustered to a certain extent.

These differences in clustering patterns likely arise from the inherent differences in the modalities and data structures of audio and image data.

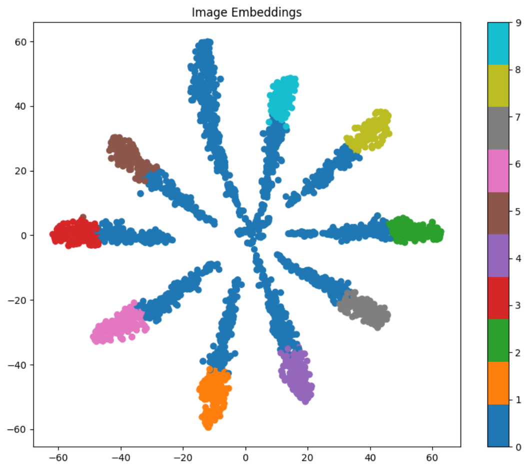

Figure 2: Embeddings from image encoder

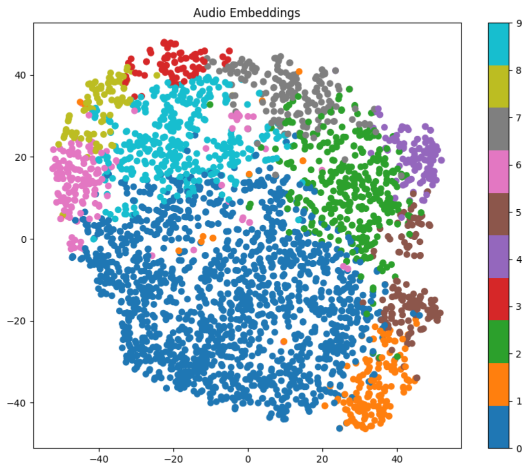

Figure 3: Embeddings from audio encoder

4 Conclusion

Incorporating multi-headed kernel to the audio encoder aided in learning more diverse relationships present in the data. The model was able to effectively capture features at different resolutions or scales. This is particularly useful for audio data, which often contains patterns and structures at various time scales. Instead of using a single convolutional layer followed by stacking more layers, the multi-headed kernel approach allows the model to learn different feature representations in parallel through the separate convolutional branches. Also, having different kernel sizes for the conv layers of image encoder worked better. Tuning the model to have the right number of dropout layers is also important, experimenting with more dropout layers regularized the model more in comparison to having two dropout layers which was consistent in producing models that were robust.

Potential measures to improve the model performance would include playing around with the number of dropout layers, number of convolutional layers and the kernel sizes for each layer.

References

[KRM19] Lyes Khacef, Laurent Rodriguez, and Benoit Miramond. “Written and spoken digits database for multimodal learning”. In: (2019).

[LA22] Rana A Lateef and Ayad R Abbas. “Tuning the hyperparameters of the 1D CNN model to improve the performance of human activity recognition”. In: *Engineering and Technology Journal* 40.04 (2022), pp. 547–554.