# AUTOMATED SPEECH RECOGNITION

## Task1

### Model Design

In alignment with the task requirements, a deep convolutional neural network (CNN) was constructed:

- Input Layer: Accepts speech image data of dimensions 98x50x1.
- Convolutional Layers: Four layers with varying filter sizes (5x5 and 7x7) and depths (128 and 256), employing 'same' padding to retain dimensionality after convolution.
- Batch Normalization Layers: Applied after each convolutional layer to standardize the inputs to the next layer, helping to stabilize and accelerate the training process.
- ReLU Activation: Employed post-batch normalization to introduce non-linearity.
- Max Pooling Layers: Following the ReLU activation to reduce spatial dimensions, hence, controlling overfitting.
- Fully Connected Layers: Two dense layers with ReLU activation and dropout to prevent overfitting.
- Output Layer: A softmax layer followed by a classification layer corresponding to the number of classes in the dataset.

### Training Procedure

- Data Preparation: Speech image data were normalized and resized, and labels were converted to categorical.
- Configuration: The Adam optimizer was utilized with an initial learning rate of 0.001. Training was conducted for 30 epochs with a mini-batch size of 64. The network was trained on a GPU to expedite the process.
- Progress Monitoring: A 'training-progress' plot was included to visualize accuracy and loss over epochs.

### Evaluation and Results

- Accuracy: The model achieved a validation accuracy of 65.34% and f1 score of 0.63, surpassing the baseline requirement.
- Loss: Training and validation loss were monitored, showing convergence indicative of a well-fitting model.
- Confusion Matrix: Generated to visualize the model’s performance across different classes.

## Task2

### Methodology

A random search was employed to efficiently traverse the hyperparameter space. The parameters varied included the number of convolutional layers (2, 3, or 4) and the number of filters per layer (32, 64, 128, or 256). Filter sizes were also chosen randomly between 3x3 and 5x5 for each layer. The search was carried out over 12 iterations to identify the combination that yielded the highest validation accuracy.

### Search Implementation

- Dynamic Layer Construction: The convolutional neural network layers were built dynamically during each iteration of the search. Layers included convolutional, batch normalization, ReLU activation, and max pooling layers, followed by fully connected layers.
- Training Options: The Adam optimizer was utilized with a learning rate of 0.001, 30 epochs for each iteration, and a mini-batch size of 64. The models were evaluated using validation data every 30 epochs.

**Results**

- Best Model Structure: The best-performing model comprised four convolutional layers, each with 32 filters, and was augmented with batch normalization, ReLU activation, and max pooling. The final layers included a fully connected layer with 256 neurons, a dropout layer, another fully connected layer with 12 neurons, and a softmax output layer.
- Performance: The optimized model reached a peak validation accuracy of 68.49%, demonstrating the effectiveness of the search strategy.

## Task3

**Model Averaging Strategy**

To improve model generalization and mitigate overfitting, a model averaging approach was employed. Five models were trained with different randomized subsets of the training data. The subsets were obtained by splitting the original training set, keeping 60% of the images for each model. This ensured that each model received a unique portion of the data, which helps to diversify the learned features and reduce variance.

**Evaluation and Results**

- Model Averaging: The predictions of the five models were combined through a majority voting scheme. For each validation image, the class with the highest number of votes was chosen as the final prediction.
- Accuracy: The model averaging approach achieved a commendable accuracy of 75.32% and f1 score of 0.764, which is a significant improvement over the single-model approach.
- Confusion Matrix: A confusion matrix was constructed to visualize the classification performance across the different speech commands.

## Task4

**Objective**

The goal of Task 4 was to perform hyperparameter optimization for our speech recognition model. Specifically, we focused on optimizing the learning rate, mini-batch size, and the number of convolutional layers and filters per layer using Bayesian optimization.

**Bayesian Optimization Results**

Bayesian optimization was employed to efficiently navigate the hyperparameter space. This approach uses a probabilistic model to predict the performance of the model with different hyperparameters and selects new hyperparameters to test by balancing exploration and exploitation.

**Learning Rate and Mini-Batch Size**

Initial optimization focused on finding the best learning rate and mini-batch size for the Adam optimizer.

- The search space for the learning rate was between $1e{-}5$ and $1e{-}2$, using a logarithmic scale to give more granularity to smaller learning rates which are typically more effective.
- The mini-batch size varied from 16 to 128, also on a logarithmic scale.

The optimal values found for these parameters were a learning rate of 0.001 and a mini-batch size of 64.

**Number of Layers**

Next, we optimized for the number of convolutional layers, where the range was set between 2 and 7 layers. The model with 5 layers was found to have the best performance.

**Filter Sizes and Numbers**

Finally, we optimized the sizes and numbers of filters in each convolutional layer.

- Filter sizes were searched within [1, 4] mapping to actual sizes [3, 5, 7, 9].
- The number of filters per layer was chosen from [1, 5], which corresponded to actual numbers [16, 32, 64, 128, 256].

The Bayesian optimization model suggested the best hyperparameters to be:

- Filter Sizes: 5, 7, 5, 7, 5
- Filter Numbers: 128, 256, 128, 128, 16

**Evaluation**

The Bayesian optimized model demonstrated an improved accuracy, validating the effectiveness of the selected hyperparameters. The optimization process balanced the model's capacity with computational efficiency, leading to a more generalizable and robust model for speech recognition tasks.

## Task5

**Methodology**

- **Early Stopping**
- Early stopping was integrated into the training process as a form of regularization to prevent overfitting. The `ValidationPatience` parameter was set to 5, meaning that the training would stop if the validation loss did not improve for five consecutive epochs. This technique helps in ensuring that the model generalizes well to unseen data by halting the training process before it learns the noise in the training data.
- **Data Augmentation**

- Data augmentation was utilized to artificially increase the diversity of the training dataset without collecting new data. The `preprocessImage` function was crafted to standardize the data, introduce random Gaussian noise, and adjust the brightness. These transformations simulate various acoustic conditions and recording qualities, making our model robust to such variations in real-world scenarios.
- **Learningrate Decay**
- Learning rate decay was implemented as part of the optimization strategy to mitigate overfitting. The decay schedule was designed to reduce the learning rate by a factor of 0.5 after every 5 epochs. This approach ensures fine-tuning of model weights and encourages convergence by gradually taking smaller steps in the weight space, thus preventing the model from overfitting to the training data's idiosyncrasies.

**Results**

- Accuracy: By employing early stopping and data augmentation, the model's accuracy on the validation set improved to 79.36%. Average F1 Score improved to 0.803.
- Training efficiency: By using early stopping, the training time decrease from 11min 13s to 9min 11s.
- Confusion Matrix: The confusion matrix revealed more balanced classification across different speech commands, indicating better generalization.

# Appendix