ASC61011 Deep Learning

Course Work – Speech Recognition

Ankur Singh Gulia

Registration Number: - 220181437

1. Introduction to Coursework

The data set provided is divided into two set of data one of which is the data used to train the model is called "Training data" and the other one is the validation set used for validating a model produced based on the data from the training data. The data represents the speech recognition patterns.

The data presented in both the sets are divided in 12-word categories suggesting different parts of the dataset such as background, yes, no etc with a total of 1661 images for training set and 1111 for the v

This coursework aims to present a domain analysis of the data and further associate the analysis with producing a baseline model. With the understanding of the baseline model, increasing the validation accuracy with the help of grid search. After the confirmation of parameters, including them in the baseline model to achieve more accuracy. Also, the creation of the depth-wise model to reduce the parameters while trying to maintain the accuracy.

Once the parameters are confirmed, cross-validation of the baseline model is done by model averaging. With the application of model averaging, it is meant to improve the validation or the in general terms the prediction of the said model.

2. Domain Analysis – Speech Recognition

It is said that like snowflakes, no two words produce speech patterns that are alike. Each word has its characteristics that helps differentiate between the words. The initial stage of data provided was already pre-processed into the form of images. The images provided were in the form of mono tone or, in technical terms, grayscale.

As mentioned earlier, the data set provided was divided into two sections:

- Training Set
  This data set was further divided into 12-word categories with a total of 2001 images in the same grayscale format. As the name suggest, this dataset is used to train the model that will be developed.

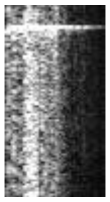


Figure 1 Raw Data - Training Set

- Validation Set
  Similar to the training set, this set is divided into 12-word categories but with a total image of 1111 in the exact same format. This format similarity is very help while validating the model developed to get a better understanding of where the model stands in term of its accuracy.

Figure 2 Raw Data - Validation set

A model generated is judged based on the accuracy to predict the word. To understand the development of the model. The initial step to achieving the goal is by building a baseline model. Once the baseline model is developed, a grid search optimization would provide the best parameter for the model developed. But to achieve more optimized output, a Model based on Depth-wise is created to understand if the parameters would be optimized further. Once this is complete, a basic model averaging script is created to optimize the accuracy of the system.

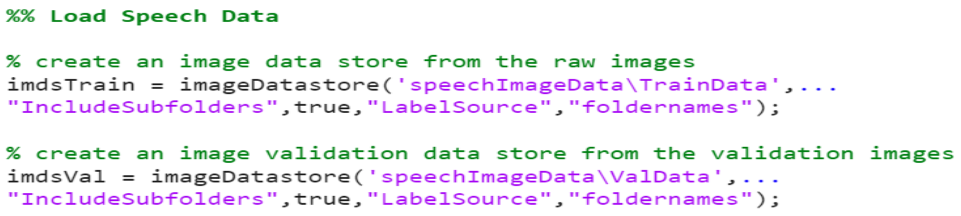
3. Baseline Model

A baseline model is a rudimentary model used as a starting point or a bench mark for the more complex model as it provides a standard based on which the models are further developed in terms of output, in this case accuracy.

The essentials for creating a baseline model is that it should be able to interpret the images provided and be able to consume the data for the prediction.

To understand the model in depth, it is better to move forward with a step by step approach rather than summarizing the model. This would be rather helpful later on while developing models that are more complex.

- The start of any model is done by providing the data to the model. This is done by using the following instructions:



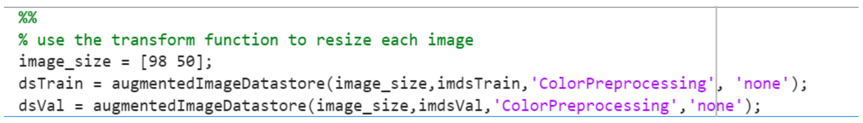
```
%% Load Speech Data

% create an image data store from the raw images
imdsTrain = imageDatastore('speechImageData\TrainData',...
"IncludeSubfolders",true,"LabelSource","foldernames");

% create an image validation data store from the validation images
imdsVal = imageDatastore('speechImageData\ValData',...
"IncludeSubfolders",true,"LabelSource","foldernames");
```

Figure 3 Loading Speech Data

As the script suggests that the dataset provided has been divided into sets which will act as training and validation set. So, it helps in reducing the effort to divide it on our own.

Once the dataset is procured, it is assigned to objects by the name of “imdsTrain” and “imdsVal” with the help of the “imageDatastore” function which in turn are used to feed the training and validation data to the machine learning model.

Next step for the model is to resize the images provided. This is done to ensure that all the images in the data set are of the same size. This allows the machine learning model to learn a consistent representation of the data and generalize well to new data. This is done with the help of the following script:



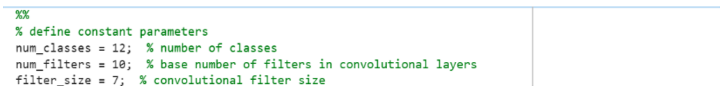
```
%%
% use the transform function to resize each image
image_size = [98 50];
dsTrain = augmentedImageDatastore(image_size,imdsTrain,'ColorPreprocessing', 'none');
dsVal = augmentedImageDatastore(image_size,imdsVal,'ColorPreprocessing','none');
```

Figure 4 Resizing and Color Pre-Processing

Apart from the resizing the images, this block of the script also suggests that there will be no color pre-processing through the model. The newly resized images have been stored in the object “dsTrain” and “dsVal”.

- Since the data pre-processing is complete. it is essential to now define parameters such as

number of filters, number of classes and filter size. So, this makes the next step for the model with the following script:

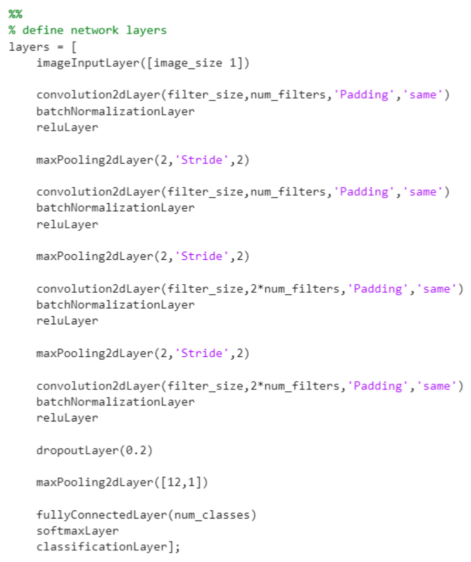


```
%%
% define constant parameters
num_classes = 12;  % number of classes
num_filters = 10;  % base number of filters in convolutional layers
filter_size = 7;  % convolutional filter size
```

Figure 5 Parameters for the Model

The number of classes was predefined to be 12 while the number of filters for the based model has been settled at 8 initially to provide a good bench mark for the more complex models. While the filter size was set at 4.

Now the model's basic necessities are complete, it is time to provide the architecture of the model based on which the model will be trained. This essentially means that the layers to train the model are scripted as:



```
%%
% define network layers
layers = [
    imageInputLayer([image_size 1])

    convolution2dLayer(filter_size,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(filter_size,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(filter_size,2*num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(filter_size,2*num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    dropoutLayer(0.2)

    maxPooling2dLayer([12,1])

    fullyConnectedLayer(num_classes)
    softmaxLayer
    classificationLayer];
```

Figure 6 Layers

In the figure 6, it clearly mentions the layers used to train the model. This block of script represents the Convolutional neural network-based architecture:

Image Input Layer: This is the initial layer that takes in the images. Image size variable indicates the size of the images while the number suggests that the images used are grayscale.

Convolution 2d Layer: This layer help in extracting features by applying set of filters to the input images. The two parameters have already been defined above. The padding parameter is set to same to ensure that the output of this operation is of same size of the input.

Batch Normalization layer: This layer normalizes the output of the previous layer to improve stability and speed of training.

Relu Layer: This layer applies the rectified linear unit activation function to the previous layer output. It is a commonly used layer in CNN that helps introducing nonlinearity to the model.

Max Pooling Layer: This layer down-samples the output of the previous layer by taking the maximum value within of size 2x2. The parameter "stride" is kept at 2 which means that the window moves 2 pixels at a time.

Drop Out Layer: This layer drops out 20% of the neurons from the previous layer to prevent overfitting.

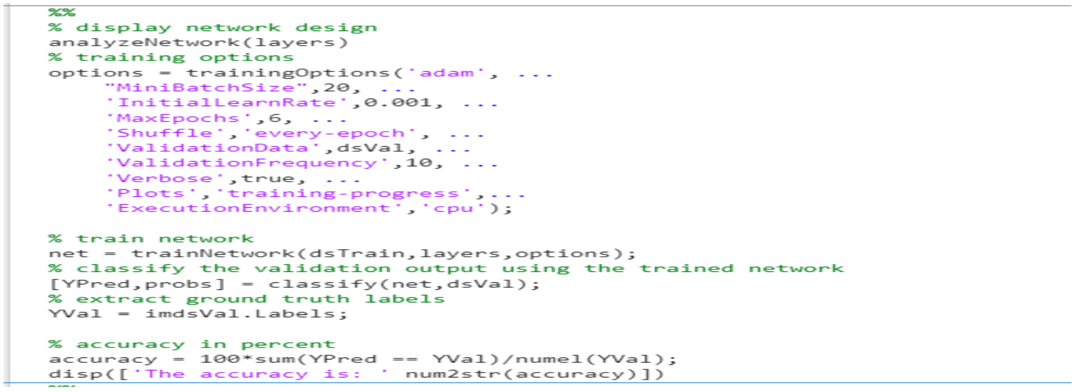
Max Pooling Layer [12,1]: Max pooling operation is performed on the previous layer's output which reduces the spatial dimensionality of the feature map while retaining important features. The [12,1] defines the pooling window size in each dimension.

Fully Connected Layer: This layer connects the neurons from the previous layer to the output layer where "num_classes" neurons corresponding to the number of classes in the classification task.

Soft Max Layer: This layer helps convert the output values into probabilities that sum upto one.

Classification layer: This layer calculates the cross-entropy loss between the predicted probabilities and the truth and updates the parameter. During testing, the outcome is given by the highest predicted probability.

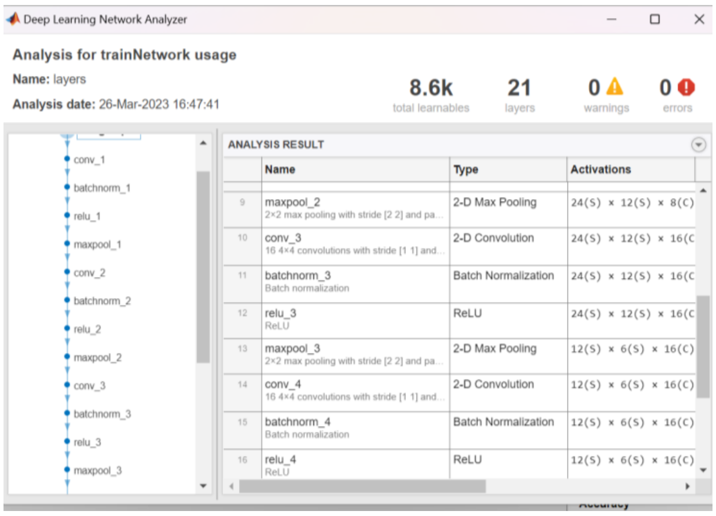
- Once the layers are defined, next is to define the environment in the model is to be trained and decide how the outcome to showcased.



```
%%
% display network design
analyzeNetwork(layers)
% training options
options = trainingOptions('adam', ...
    "MiniBatchSize",20, ...
    'InitialLearnRate',0.001, ...
    'MaxEpochs',6, ...
    'Shuffle','every-epoch', ...
    'ValidationData',dsVal, ...
    'ValidationFrequency',10, ...
    'Verbose',true, ...
    'Plots','training-progress',...
    'ExecutionEnvironment','cpu');

% train network
net = trainNetwork(dsTrain,layers,options);
% classify the validation output using the trained network
[YPred,probs] = classify(net,dsVal);
% extract ground truth labels
YVal = imdsVal.Labels;

% accuracy in percent
accuracy = 100*sum(YPred == YVal)/numel(YVal);
disp(['The accuracy is: ' num2str(accuracy)])
```

Figure 7 Training Environment

With the help of the training options, the model is complete as it sets up the environment of the model. With this, the baseline model is complete. Now to reach a standard accuracy for prediction, the model is trained with a variation of parameter and different combination of the layers.



Analysis for trainNetwork usage
Name: layers
Analysis date: 26-Mar-2023 16:47:41

8.6k total learnables | 21 layers | 0 warnings | 0 errors

ANALYSIS RESULT

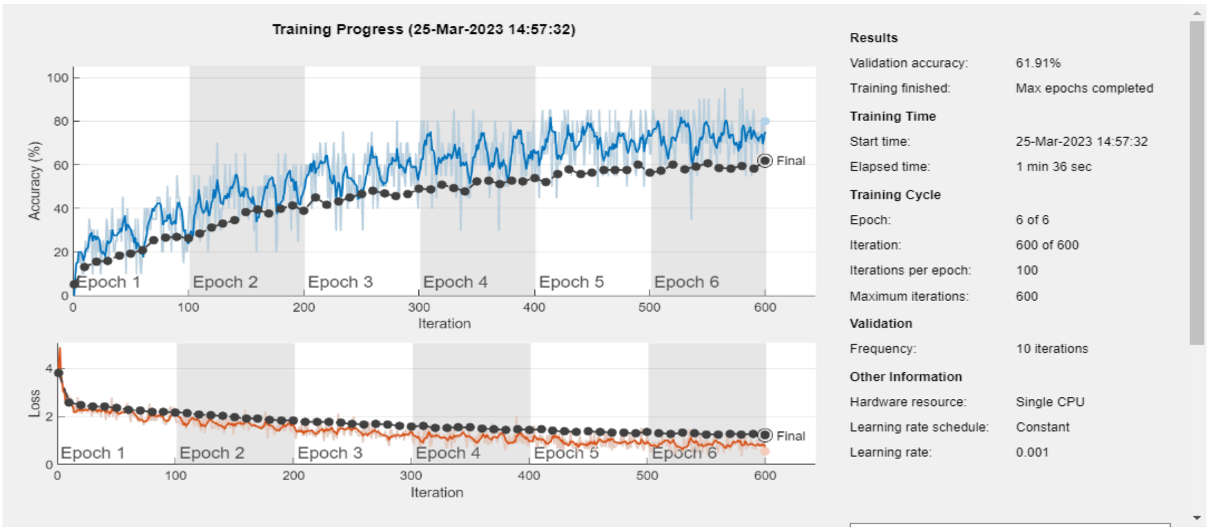
	Name	Type	Activations
9	maxpool_2 2×2 max pooling with stride [2 2] and pa...	2-D Max Pooling	24(S) × 12(S) × 8(C)
10	conv_3 16 4×4 convolutions with stride [1 1] and...	2-D Convolution	24(S) × 12(S) × 16(C
11	batchnorm_3 Batch normalization	Batch Normalization	24(S) × 12(S) × 16(C
12	relu_3 ReLU	ReLU	24(S) × 12(S) × 16(C
13	maxpool_3 2×2 max pooling with stride [2 2] and pa...	2-D Max Pooling	12(S) × 6(S) × 16(C)
14	conv_4 16 4×4 convolutions with stride [1 1] and...	2-D Convolution	12(S) × 6(S) × 16(C)
15	batchnorm_4 Batch normalization	Batch Normalization	12(S) × 6(S) × 16(C)
16	relu_4 ReLU	ReLU	12(S) × 6(S) × 16(C)

Figure 8 DL Network Analyser

Finally, after several trial and error, a benchmark accuracy has been reached with the following things to keep in mind:

- Parameters:
  1. Number of filters: 8
  2. Filter Size: 4
  3. Number of set of layers: 4

These parameters helped reach an accuracy of the model to 61% as shown in figure 8.



Results

Validation accuracy:	61.91%
Training finished:	Max epochs completed

Training Time

Start time:	25-Mar-2023 14:57:32
Elapsed time:	1 min 36 sec

Training Cycle

Epoch:	6 of 6
Iteration:	600 of 600
Iterations per epoch:	100
Maximum iterations:	600

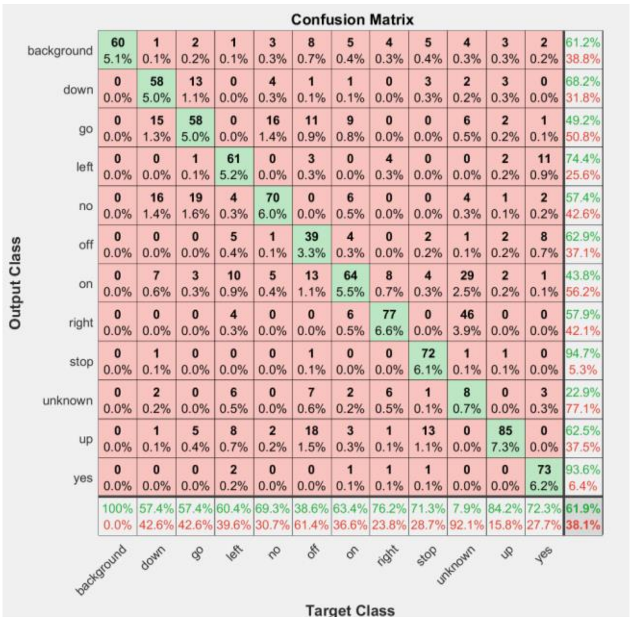
Validation

Frequency:	10 iterations

Other Information

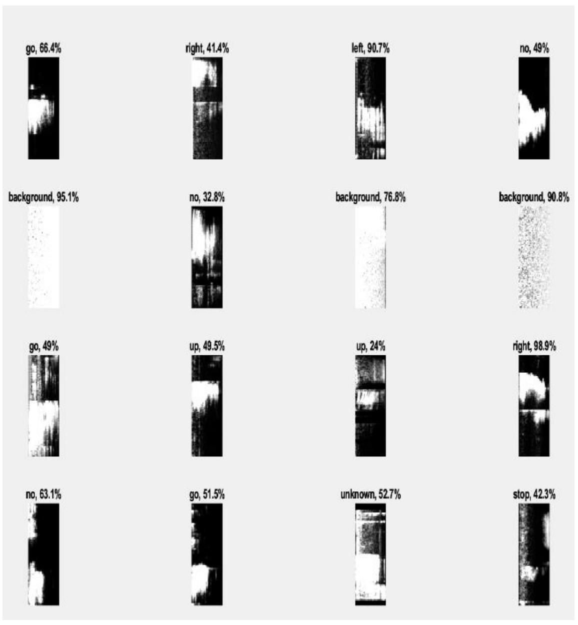
Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.001

Figure 9 Baseline Model Output



Confusion Matrix

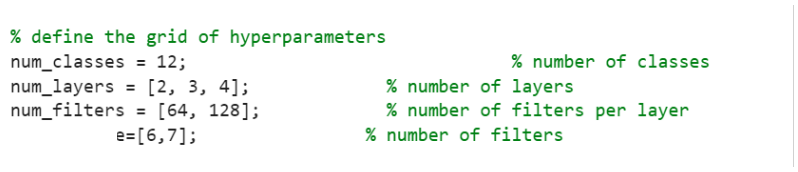
Output Class \ Target Class	background	down	go	left	no	off	on	right	stop	unknown	up	yes	
background	60 5.1%	1 0.1%	2 0.2%	1 0.1%	3 0.3%	8 0.7%	5 0.4%	4 0.3%	5 0.4%	4 0.3%	3 0.3%	2 0.2%	61.2% 38.8%
down	0 0.0%	58 5.0%	13 1.1%	0 0.0%	4 0.3%	1 0.1%	1 0.1%	0 0.0%	3 0.3%	2 0.2%	3 0.3%	0 0.0%	68.2% 31.8%
go	0 0.0%	15 1.3%	58 5.0%	0 0.0%	16 1.4%	11 0.9%	9 0.8%	0 0.0%	0 0.0%	6 0.5%	2 0.2%	1 0.1%	49.2% 50.8%
left	0 0.0%	0 0.0%	1 0.1%	61 5.2%	0 0.0%	3 0.3%	0 0.0%	4 0.3%	0 0.0%	0 0.0%	2 0.2%	11 0.9%	74.4% 25.6%
no	0 0.0%	16 1.4%	19 1.6%	4 0.3%	70 6.0%	0 0.0%	6 0.5%	0 0.0%	0 0.0%	4 0.3%	1 0.1%	2 0.2%	57.4% 42.6%
off	0 0.0%	0 0.0%	0 0.0%	5 0.4%	1 0.1%	39 3.3%	4 0.3%	0 0.0%	2 0.2%	1 0.1%	2 0.2%	8 0.7%	62.9% 37.1%
on	0 0.0%	7 0.6%	3 0.3%	10 0.9%	5 0.4%	13 1.1%	64 5.5%	8 0.7%	4 0.3%	29 2.5%	2 0.2%	1 0.1%	43.8% 56.2%
right	0 0.0%	0 0.0%	0 0.0%	4 0.3%	0 0.0%	0 0.0%	6 0.5%	77 6.6%	0 0.0%	46 3.9%	0 0.0%	0 0.0%	57.9% 42.1%
stop	0 0.0%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	72 6.1%	1 0.1%	1 0.1%	0 0.0%	94.7% 5.3%
unknown	0 0.0%	2 0.2%	0 0.0%	6 0.5%	0 0.0%	7 0.6%	2 0.2%	6 0.5%	1 0.1%	8 0.7%	0 0.0%	3 0.3%	22.9% 77.1%
up	0 0.0%	1 0.1%	5 0.4%	8 0.7%	2 0.2%	18 1.5%	3 0.3%	1 0.1%	13 1.1%	0 0.0%	85 7.3%	0 0.0%	62.5% 37.5%
yes	0 0.0%	0 0.0%	0 0.0%	2 0.2%	0 0.0%	0 0.0%	1 0.1%	1 0.1%	1 0.1%	0 0.0%	0 0.0%	73 6.2%	93.6% 6.4%
	100% 0.0%	57.4% 42.6%	57.4% 42.6%	60.4% 39.6%	69.3% 30.7%	38.6% 61.4%	63.4% 36.6%	76.2% 23.8%	71.3% 28.7%	7.9% 92.1%	84.2% 15.8%	72.3% 27.7%	61.9% 38.1%



4. Grid Search

Grid Search is a technique for Hyperparameter tuning used in Machine Learning models to find optimal values of hypermeters for a given model. Hyperparameters are parameters that are not learned from the training data but are set before the initial start of the training. But, since the training model developed uses a combination of hyperparameter, it usually takes a lot of time for a normal computer.

A model for grid search is similar to the baseline model up until the pre-processed images are

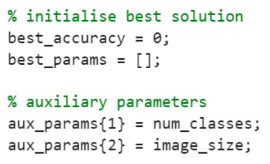


```
% define the grid of hyperparameters
num_classes = 12;                                % number of classes
num_layers = [2, 3, 4];             % number of layers
num_filters = [64, 128];            % number of filters per layer
          e=[6,7];                 % number of filters
```

Figure 10 Fine Tuning Parameters

resized. After that, parameter that must run through the model to find the best hyper tunned parameter for the model generated.

As one establishes an array for the parameters, it defines the range of the grid search. After the parameters are defined, the system initializes certain variables which are given in Figure 10.

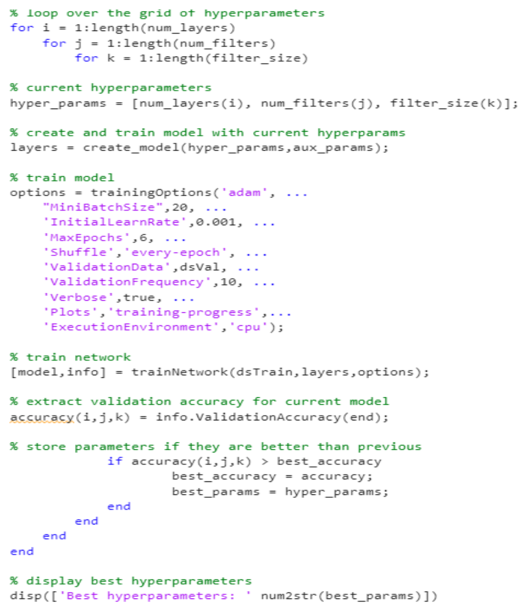


```
% initialise best solution
best_accuracy = 0;
best_params = [];

% auxiliary parameters
aux_params{1} = num_classes;
aux_params{2} = image_size;
```

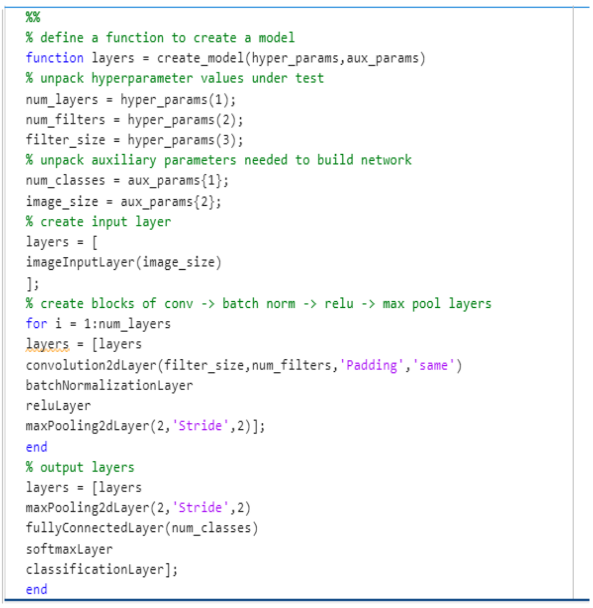
Figure 11 Initializing Values

Once these initial stages work is complete, now the main loop for the search is activated which is scripted as:

The loop provided in this situation is a nested loop for the grid search optimization with the parameters such as “num_filters”,” filter_size” and “num_layers”. This helps in determining the best



```
% loop over the grid of hyperparameters
for i = 1:length(num_layers)
    for j = 1:length(num_filters)
        for k = 1:length(filter_size)

% current hyperparameters
hyper_params = [num_layers(i), num_filters(j), filter_size(k)];

% create and train model with current hyperparams
layers = create_model(hyper_params,aux_params);

% train model
options = trainingOptions('adam', ...
    "MiniBatchSize",20, ...
    'InitialLearnRate',0.001, ...
    'MaxEpochs',6, ...
    'Shuffle','every-epoch', ...
    'ValidationData',dsVal, ...
    'ValidationFrequency',10, ...
    'Verbose',true, ...
    'Plots','training-progress',...
    'ExecutionEnvironment','cpu');

% train network
[model,info] = trainNetwork(dsTrain,layers,options);

% extract validation accuracy for current model
accuracy(i,j,k) = info.ValidationAccuracy(end);

% store parameters if they are better than previous
            if accuracy(i,j,k) > best_accuracy
                    best_accuracy = accuracy;
                    best_params = hyper_params;
            end
        end
    end
end

% display best hyperparameters
disp(['Best hyperparameters: ' num2str(best_params)])
```



```
%%
% define a function to create a model
function layers = create_model(hyper_params,aux_params)
% unpack hyperparameter values under test
num_layers = hyper_params(1);
num_filters = hyper_params(2);
filter_size = hyper_params(3);
% unpack auxiliary parameters needed to build network
num_classes = aux_params{1};
image_size = aux_params{2};
% create input layer
layers = [
imageInputLayer(image_size)
];
% create blocks of conv -> batch norm -> relu -> max pool layers
for i = 1:num_layers
layers = [layers
convolution2dLayer(filter_size,num_filters,'Padding','same')
batchNormalizationLayer
reluLayer
maxPooling2dLayer(2,'Stride',2)];
end
% output layers
layers = [layers
maxPooling2dLayer(2,'Stride',2)
fullyConnectedLayer(num_classes)
softmaxLayer
classificationLayer];
end
```

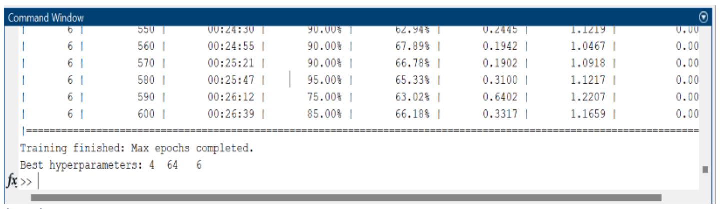
Figure 12 Grid Search Loop and Function

output while training all the models. If the accuracy increases from its previous best accuracy, then the output changes to the present hyperparameters and the accuracy.

While doing so, it also provides the environment architecture for training model. Also, in the grid search loop, the function create model on the basis of which the layers and other parameters are decided from the array is present.

Once all this is done, the initial model of the grid search is completed. The output for the parameters were given as:

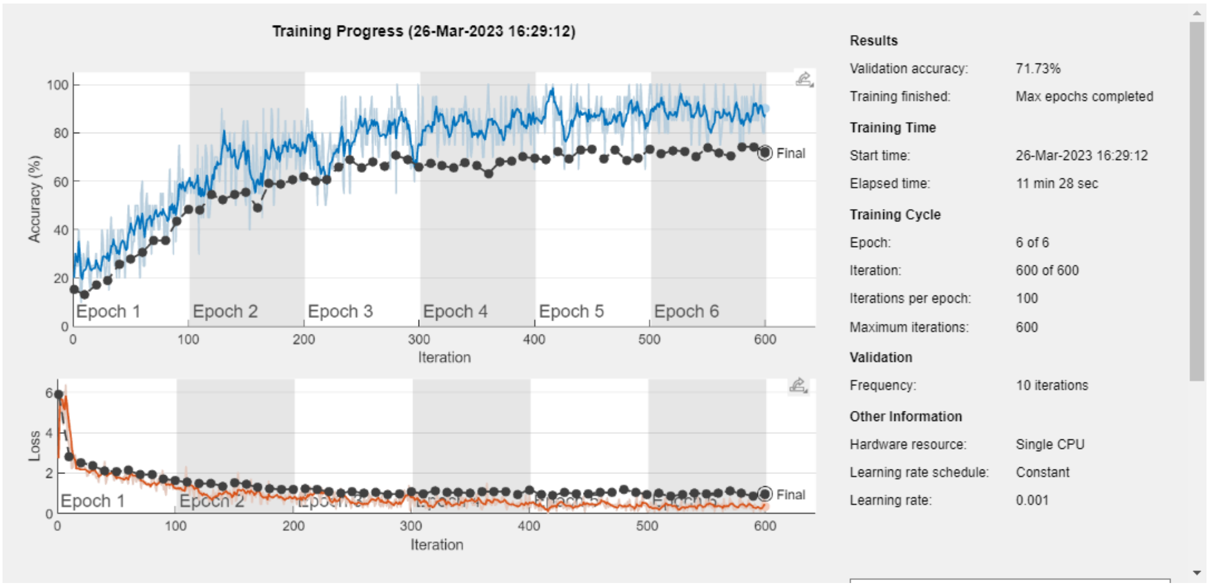
- ➢ Number of Layers: 4
- ➢ Number of Filters: 64
- ➢ Filter Size : 6



```
|        6 |      550 |     00:24:30 |      90.00% |      62.94% |      0.2445 |      1.1219 |      0.00
|        6 |      560 |     00:24:55 |      90.00% |      67.89% |      0.1942 |      1.0467 |      0.00
|        6 |      570 |     00:25:21 |      90.00% |      66.78% |      0.1902 |      1.0918 |      0.00
|        6 |      580 |     00:25:47 |      95.00% |      65.33% |      0.3100 |      1.1217 |      0.00
|        6 |      590 |     00:26:12 |      75.00% |      63.02% |      0.6402 |      1.2207 |      0.00
|        6 |      600 |     00:26:39 |      85.00% |      66.18% |      0.3317 |      1.1659 |      0.00
|=====================================================================================================
Training finished: Max epochs completed.
Best hyperparameters: 4   64    6
>>
```

Figure 13 Grid Search Output

The output with these parameters gave an accuracy of 71% as show below:



```
Results
Validation accuracy:        71.73%
Training finished:          Max epochs completed
Training Time
Start time:                 26-Mar-2023 16:29:12
Elapsed time:               11 min 28 sec
Training Cycle
Epoch:                      6 of 6
Iteration:                  600 of 600
Iterations per epoch:       100
Maximum iterations:         600
Validation
Frequency:                  10 iterations
Other Information
Hardware resource:          Single CPU
Learning rate schedule:     Constant
Learning rate:              0.001
```

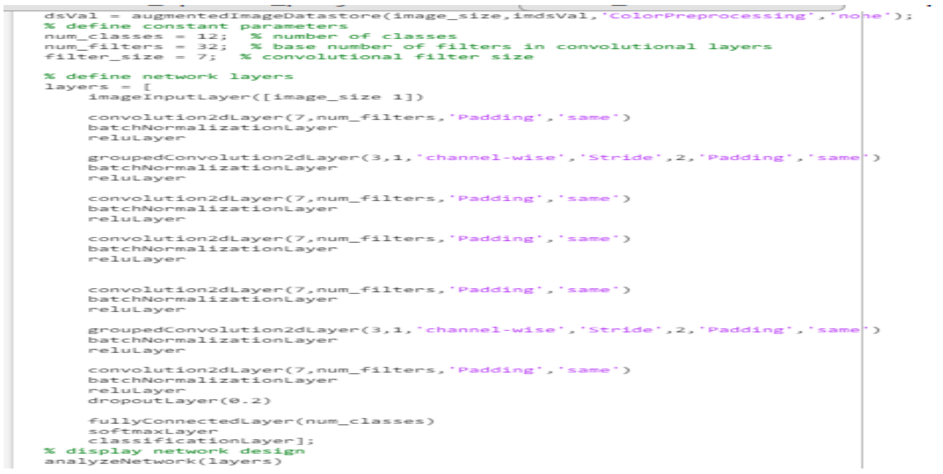
Figure 14 Grid Search Output 2

5. Depth Wise Model

A convolutional neural network where each filter is applied to a single input channel is the depth-wise model. It is essentially used to reduce the parameters that are helpful for training the model while maintaining high accuracy.

The initial set up for this model is the same as the baseline model. Essentially, the main difference comes while experimenting with the parameters so as to achieve optimal solution. Also, the main difference in this model and the baseline model is the use of grouped 2D convolutional layer. This layer can be used in situations with multiple channels in the input data, are processed independently while preserving their channel-wise relationships.

The difference is shown by figure 15 where the positioning of the layers suggests the reduction in parameters and further reduction accuracy.



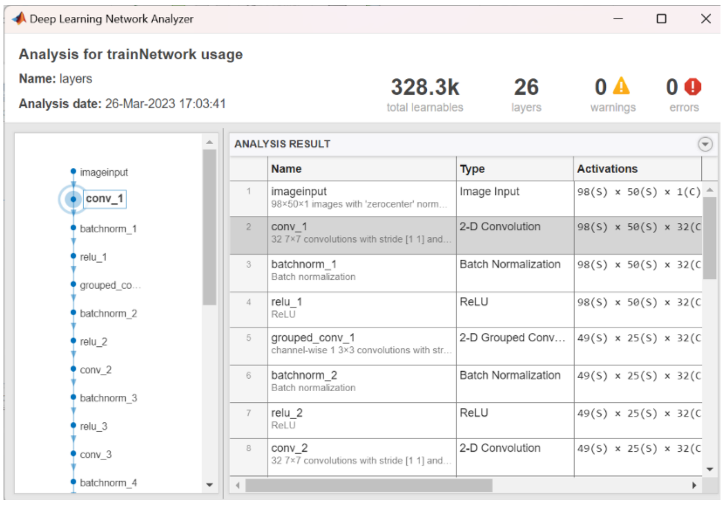
```
dsVal = augmentedImageDatastore(image_size,imdsVal,'ColorPreprocessing','none');
% define constant parameters
num_classes = 12;   % number of classes
num_filters = 32;   % base number of filters in convolutional layers
filter_size = 7;   % convolutional filter size

% define network layers
layers = [
    imageInputLayer([image_size 1])

    convolution2dLayer(7,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    groupedConvolution2dLayer(3,1,'channel-wise','Stride',2,'Padding','same')
    batchNormalizationLayer
    reluLayer

    convolution2dLayer(7,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    convolution2dLayer(7,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer


    convolution2dLayer(7,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    groupedConvolution2dLayer(3,1,'channel-wise','Stride',2,'Padding','same')
    batchNormalizationLayer
    reluLayer

    convolution2dLayer(7,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer
    dropoutLayer(0.2)

    fullyConnectedLayer(num_classes)
    softmaxLayer
    classificationLayer];
% display network design
analyzeNetwork(layers)
```

Figure 15 Depth Wise Model

As per the training of the model, it suggested that the when the parameters are reduced by a large margin the accuracy also decrease with it. Hence after careful consideration and multiple trail and errors, the outcome of the final parameters was given as:

- Number of Filters: 16
- Filter Size : 06
- Layer Pattern : C-G-C-C-C-G-C

The initial analyser shows the output and sequences of the layers as suggested in figure 16:


Deep Learning Network Analyzer

Analysis for trainNetwork usage

Name: layers

Analysis date: 26-Mar-2023 17:03:41

328.3k total learnables | 26 layers | 0 warnings | 0 errors

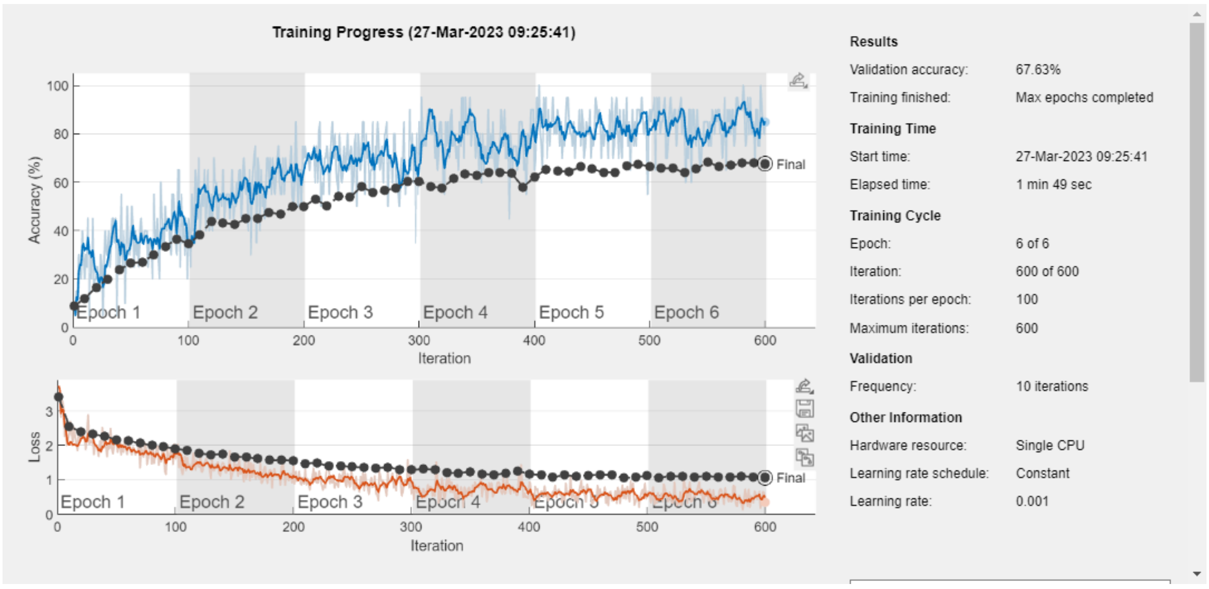
imageinput, conv_1, batchnorm_1, relu_1, grouped_co..., batchnorm_2, relu_2, conv_2, batchnorm_3, relu_3, conv_3, batchnorm_4

ANALYSIS RESULT

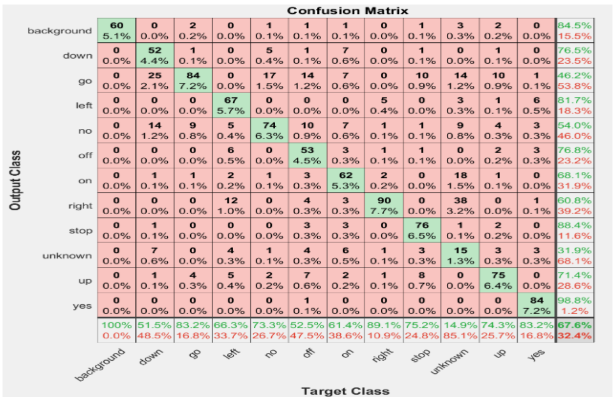
	Name	Type	Activations
1	imageinput 98×50×1 images with 'zerocenter' norm...	Image Input	98(S) × 50(S) × 1(C)
2	conv_1 32 7×7 convolutions with stride [1 1] and...	2-D Convolution	98(S) × 50(S) × 32(C
3	batchnorm_1 Batch normalization	Batch Normalization	98(S) × 50(S) × 32(C
4	relu_1 ReLU	ReLU	98(S) × 50(S) × 32(C
5	grouped_conv_1 channel-wise 1 3×3 convolutions with str...	2-D Grouped Conv...	49(S) × 25(S) × 32(C
6	batchnorm_2 Batch normalization	Batch Normalization	49(S) × 25(S) × 32(C
7	relu_2 ReLU	ReLU	49(S) × 25(S) × 32(C
8	conv_2 32 7×7 convolutions with stride [1 1] and...	2-D Convolution	49(S) × 25(S) × 32(C

Figure 16 DL Network analyser for Depth-wise

The output for the depth wise model although reduced the accuracy but helped in the reduction of the parameters, the current accuracy stands at 67%.



Results	
Validation accuracy:	67.63%
Training finished:	Max epochs completed
Training Time	
Start time:	27-Mar-2023 09:25:41
Elapsed time:	1 min 49 sec
Training Cycle	
Epoch:	6 of 6
Iteration:	600 of 600
Iterations per epoch:	100
Maximum iterations:	600
Validation	
Frequency:	10 iterations
Other Information	
Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.001


Confusion Matrix

Output Class \ Target Class	background	down	go	left	no	off	on	right	stop	unknown	up	yes	
background	60 5.1%	0 0.0%	2 0.2%	0 0.0%	1 0.1%	1 0.1%	1 0.1%	0 0.0%	1 0.1%	3 0.3%	2 0.2%	0 0.0%	84.5% 15.5%
down	0 0.0%	52 4.4%	1 0.1%	0 0.0%	5 0.4%	1 0.1%	7 0.6%	0 0.0%	1 0.1%	0 0.0%	1 0.1%	0 0.0%	76.5% 23.5%
go	0 0.0%	25 2.1%	84 7.2%	0 0.0%	17 1.5%	14 1.2%	7 0.6%	0 0.0%	10 0.9%	14 1.2%	10 0.9%	1 0.1%	46.2% 53.8%
left	0 0.0%	0 0.0%	0 0.0%	67 5.7%	0 0.0%	0 0.0%	0 0.0%	5 0.4%	0 0.0%	3 0.3%	1 0.1%	6 0.5%	81.7% 18.3%
no	0 0.0%	14 1.2%	9 0.8%	5 0.4%	74 6.3%	10 0.9%	7 0.6%	1 0.1%	1 0.1%	9 0.8%	4 0.3%	3 0.3%	54.0% 46.0%
off	0 0.0%	0 0.0%	0 0.0%	6 0.5%	0 0.0%	53 4.5%	3 0.3%	1 0.1%	1 0.1%	0 0.0%	2 0.2%	3 0.3%	76.8% 23.2%
on	0 0.0%	1 0.1%	1 0.1%	2 0.2%	1 0.1%	3 0.3%	62 5.3%	2 0.2%	0 0.0%	18 1.5%	1 0.1%	0 0.0%	68.1% 31.9%
right	0 0.0%	0 0.0%	0 0.0%	12 1.0%	0 0.0%	4 0.3%	3 0.3%	90 7.7%	0 0.0%	38 3.2%	0 0.0%	1 0.1%	60.8% 39.2%
stop	0 0.0%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	3 0.3%	3 0.3%	0 0.0%	76 6.5%	1 0.1%	2 0.2%	0 0.0%	88.4% 11.6%
unknown	0 0.0%	7 0.6%	0 0.0%	4 0.3%	1 0.1%	4 0.3%	6 0.5%	1 0.1%	3 0.3%	15 1.3%	3 0.3%	3 0.3%	31.9% 68.1%
up	0 0.0%	1 0.1%	4 0.3%	5 0.4%	2 0.2%	7 0.6%	2 0.2%	1 0.1%	8 0.7%	0 0.0%	75 6.4%	0 0.0%	71.4% 28.6%
yes	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	84 7.2%	98.8% 1.2%
	100% 0.0%	51.5% 48.5%	83.2% 16.8%	66.3% 33.7%	73.3% 26.7%	52.5% 47.5%	61.4% 38.6%	89.1% 10.9%	75.2% 24.8%	14.9% 85.1%	74.3% 25.7%	83.2% 16.8%	67.6% 32.4%

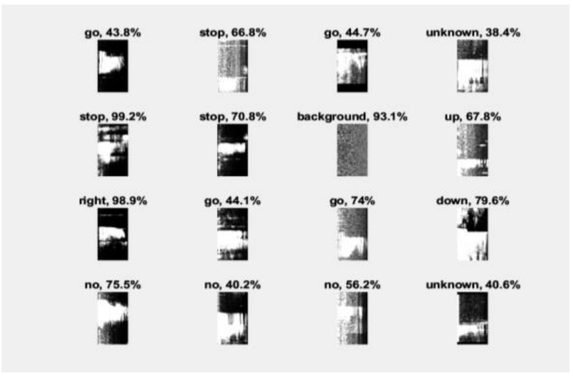


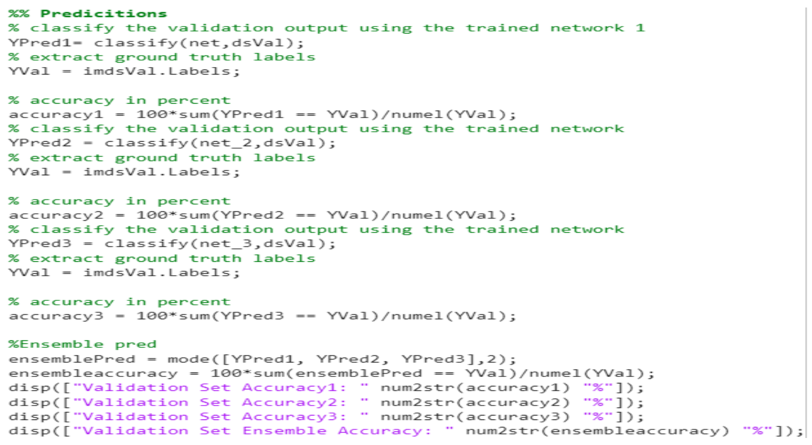
Figure 17 Depth Wise Output

6. Model Averaging

For a Machine learning, it is a technique used to improve the predicted performance of the models by combining multiple models together. It is preferable used in situations where the models are used to capture all the patterns and relationships in the data.

Here, same dataset is used with different parameters or different algorithms. Finally, when the training is complete, the prediction are then combined to form a final output which is typically more accurate and robust than any individual model.

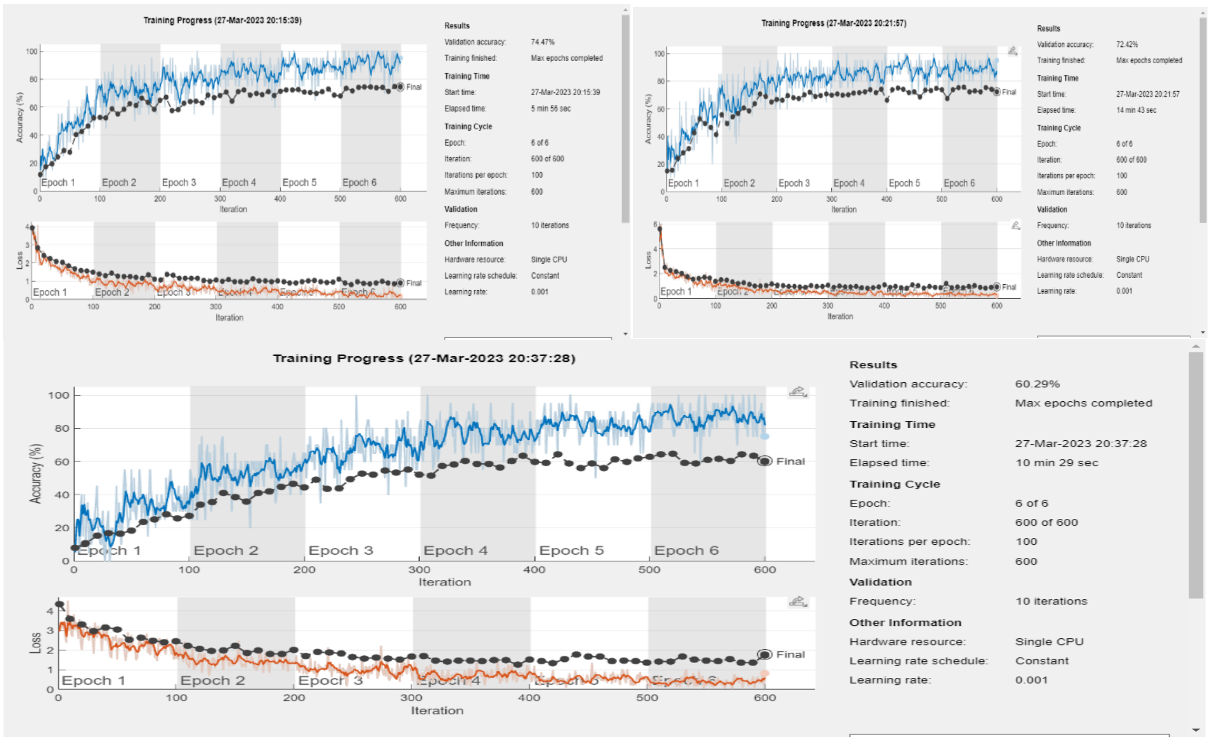
Essentially after training three different model with the same dataset, we use the following script for combining the accuracy:



```
%% Predicitions
% classify the validation output using the trained network 1
YPred1= classify(net,dsVal);
% extract ground truth labels
YVal = imdsVal.Labels;

% accuracy in percent
accuracy1 = 100*sum(YPred1 == YVal)/numel(YVal);
% classify the validation output using the trained network
YPred2 = classify(net_2,dsVal);
% extract ground truth labels
YVal = imdsVal.Labels;

% accuracy in percent
accuracy2 = 100*sum(YPred2 == YVal)/numel(YVal);
% classify the validation output using the trained network
YPred3 = classify(net_3,dsVal);
% extract ground truth labels
YVal = imdsVal.Labels;

% accuracy in percent
accuracy3 = 100*sum(YPred3 == YVal)/numel(YVal);

%Ensemble pred
ensemblePred = mode([YPred1, YPred2, YPred3],2);
ensembleaccuracy = 100*sum(ensemblePred == YVal)/numel(YVal);
disp(["Validation Set Accuracy1: " num2str(accuracy1) "%"]);
disp(["Validation Set Accuracy2: " num2str(accuracy2) "%"]);
disp(["Validation Set Accuracy3: " num2str(accuracy3) "%"]);
disp(["Validation Set Ensemble Accuracy: " num2str(ensembleaccuracy) "%"]);
```

Figure 18 Combining Model Accuracy

Before this is done the accuracys separately are shown as:



Training Progress (27-Mar-2023 20:15:39)
Training Progress (27-Mar-2023 20:21:57)
Training Progress (27-Mar-2023 20:37:28)
Accuracy (%)
Loss
Iteration
Epoch 1
Epoch 2
Epoch 3
Epoch 4
Epoch 5
Epoch 6
Final

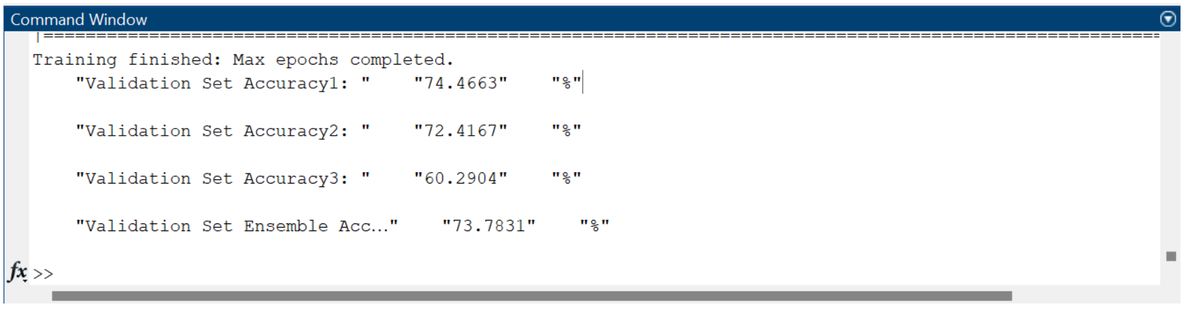
```
Results
Validation accuracy:        74.47%
Training finished:          Max epochs completed
Training Time
Start time:                 27-Mar-2023 20:15:39
Elapsed time:               5 min 56 sec
Training Cycle
Epoch:                      6 of 6
Iteration:                  600 of 600
Iterations per epoch:       100
Maximum iterations:         600
Validation
Frequency:                  10 iterations
Other Information
Hardware resource:          Single CPU
Learning rate schedule:     Constant
Learning rate:              0.001
```

```
Results
Validation accuracy:        72.42%
Training finished:          Max epochs completed
Training Time
Start time:                 27-Mar-2023 20:21:57
Elapsed time:               14 min 43 sec
Training Cycle
Epoch:                      6 of 6
Iteration:                  600 of 600
Iterations per epoch:       100
Maximum iterations:         600
Validation
Frequency:                  10 iterations
Other Information
Hardware resource:          Single CPU
Learning rate schedule:     Constant
Learning rate:              0.001
```

```
Results
Validation accuracy:        60.29%
Training finished:          Max epochs completed
Training Time
Start time:                 27-Mar-2023 20:37:28
Elapsed time:               10 min 29 sec
Training Cycle
Epoch:                      6 of 6
Iteration:                  600 of 600
Iterations per epoch:       100
Maximum iterations:         600
Validation
Frequency:                  10 iterations
Other Information
Hardware resource:          Single CPU
Learning rate schedule:     Constant
Learning rate:              0.001
```

Figure 19 Initial Training for models

The final result is shown based on the script as:



```
|=====================================================================================
Training finished: Max epochs completed.
    "Validation Set Accuracy1: "    "74.4663"    "%"

    "Validation Set Accuracy2: "    "72.4167"    "%"

    "Validation Set Accuracy3: "    "60.2904"    "%"

    "Validation Set Ensemble Acc…"    "73.7831"    "%"

fx >>
```

Figure 20 Model Averaging - Result

With this a general model for speech recognition using the scripts have been develop. As you can see that it is difficult to train the models as they get complex and even more complex to design the models.

This designing issue can be resolved with the help of the model designing using the Deep network designer.

7. Deep Network Designer – Resnet

A graphical user interface (GUI) tool called deep Network Designer in MATLAB's Deep Learning Toolbox enables users to create, visualize, and change deep neural networks without having to have a thorough understanding of deep learning.

For this case, a model based on Resnet was developed to improve the model's accuracy. This was essentially done by doing trial and error of using different layers. Here, a Resnet model was used by skipping some layers to obtain a good accuracy.

The resulting output was show below including a Resnet design and output showing the training accuracy as well.

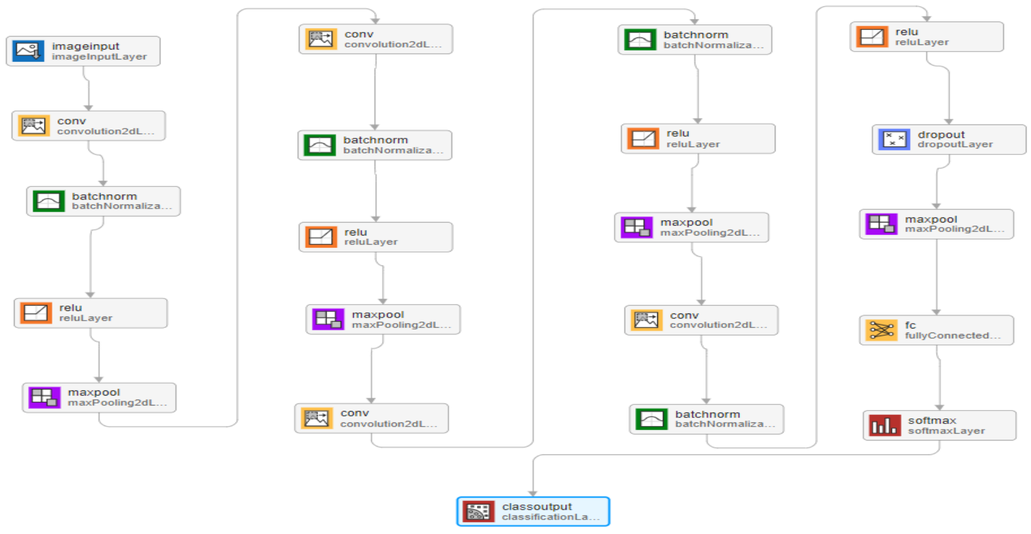


Figure 21 Design for Layers

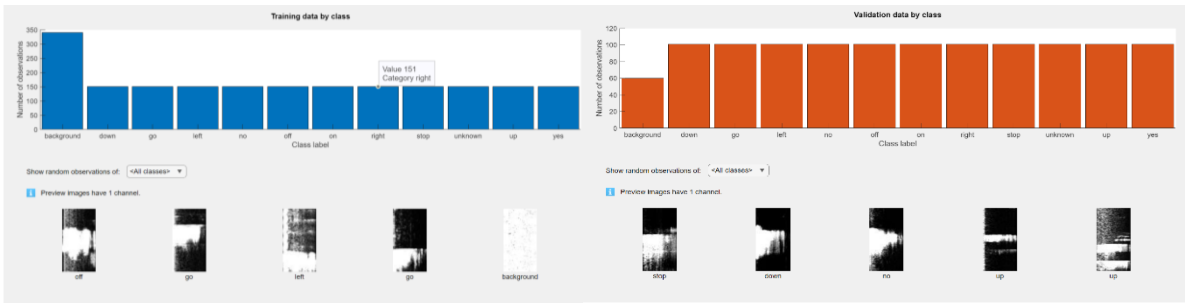
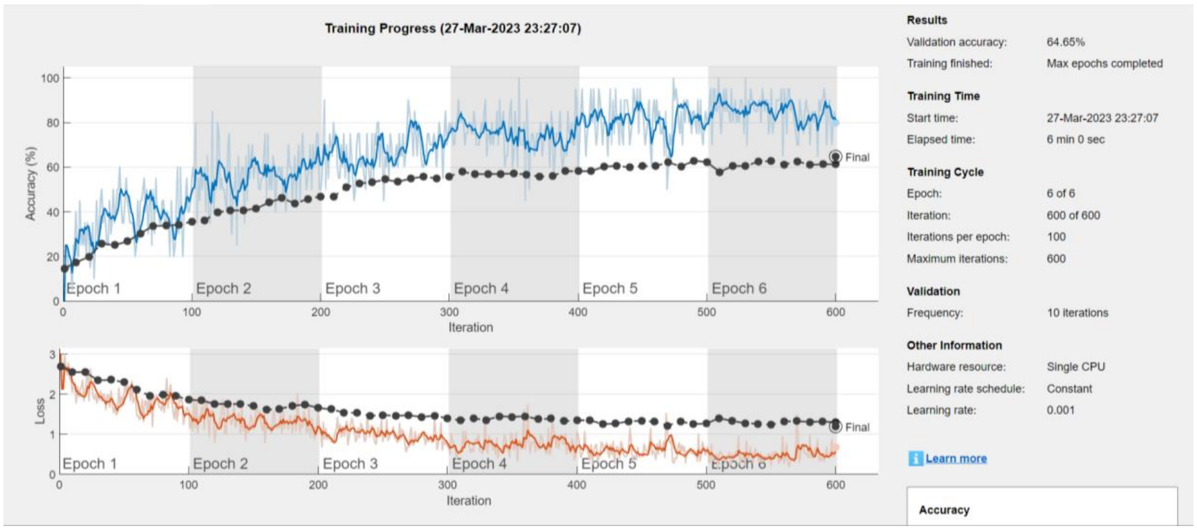


Figure 22 Training and Validation Sets



Results

Validation accuracy:	64.65%
Training finished:	Max epochs completed

Training Time

Start time:	27-Mar-2023 23:27:07
Elapsed time:	6 min 0 sec

Training Cycle

Epoch:	6 of 6
Iteration:	600 of 600
Iterations per epoch:	100
Maximum iterations:	600

Validation

Frequency:	10 iterations

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.001

Figure 23 Resnet Based Model

The above plots give a glimpse of a Machine learning Model developed on a Deep Network Designer.

8. Conclusion

This course work consisted of creation of a complete pipeline for a data set based on study of 12 different patterns for speech recognition. The process started with the domain analysis of the dataset and Speech recognition. Once the domain analysis was complete. It gave enough information to construct a baseline model based on the 12 different patterns for prediction.

This process was complete when the baseline model achieved required accuracy. Now, the model needs to be improved by the help of the grid search model. The grid search model provided the best hyper parameter to take out the best accuracy.

From here on, a model was developed that was used to determine whether further parameters can be reduced to achieve the optimal solution was developed and the model was called depth-wise.

Upon completion of the task above, another model was developed, it was suggested that a model averaging technique should be used to find the average accuracy that is generally a better and robust solution.Now, a model based on the deep Network designer was prepared to understand the output of the system.

With the completion of a machine learning model, the classification was done which helped in creating an effecting ML Model.

Code Appendix

Baseline Model

```
clear

% define the random number seed for repeatable results
rng(1,'twister');

%% Load Speech Data

% create an image data store from the raw images
imdsTrain = imageDatastore('speechImageData\TrainData',...
"IncludeSubfolders",true,"LabelSource","foldernames");

% create an image validation data store from the validation images
imdsVal = imageDatastore('speechImageData\ValData',...
"IncludeSubfolders",true,"LabelSource","foldernames");

%%
% use the transform function to resize each image
image_size = [98 50];
dsTrain = augmentedImageDatastore(image_size,imdsTrain,'ColorPreprocessing',
'none');
dsVal = augmentedImageDatastore(image_size,imdsVal,'ColorPreprocessing','none');
%%
% define constant parameters
num_classes = 12;  % number of classes
num_filters = 8;  % base number of filters in convolutional layers
filter_size = 6;  % convolutional filter size
%%
% define network layers
layers = [
    imageInputLayer([image_size 1])

    convolution2dLayer(filter_size,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(filter_size,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(filter_size,2*num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(filter_size,2*num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    dropoutLayer(0.2)

    maxPooling2dLayer([12,1])
    fullyConnectedLayer(num_classes)
    softmaxLayer
    classificationLayer];
%%
% display network design
analyzeNetwork(layers)
% training options
options = trainingOptions('adam', ...
    "MiniBatchSize",20, ...
    'InitialLearnRate',0.001, ...
    'MaxEpochs',6, ...
    'Shuffle','every-epoch', ...
    'ValidationData',dsVal, ...
    'ValidationFrequency',10, ...
    'Verbose',true, ...
    'Plots','training-progress',...
    'ExecutionEnvironment','cpu');

% train network
net = trainNetwork(dsTrain,layers,options);
% classify the validation output using the trained network
[YPred,probs] = classify(net,dsVal);
% extract ground truth labels
YVal = imdsVal.Labels;

% accuracy in percent
accuracy = 100*sum(YPred == YVal)/numel(YVal);
disp(['The accuracy is: ' num2str(accuracy)])
%%
% plot confusion matrix
figure;
plotconfusion(YVal,YPred)
% Display sample test images with predicted labels and
% the predicted probabilities of the images having those labels.
idx = randperm(numel(imdsVal.Files),16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(imdsVal,idx(i));
    imshow(I)
    label = YPred(idx(i));
    title(string(label)+ ", "+num2str(100*max(probs(idx(i),:)),3)+"%");
end
disp(["Validation Set Accuracy: " num2str(accuracy) "%"]);
```

Grid Search

```
clear

% define the random number seed for repeatable results
rng(1,'twister');

%% Load Speech Data

% create an image data store from the raw images
imdsTrain = imageDatastore('speechImageData\TrainData',...
"IncludeSubfolders",true,"LabelSource","foldernames");

% create an image validation data store from the validation images
imdsVal = imageDatastore('speechImageData\ValData',...
"IncludeSubfolders",true,"LabelSource","foldernames");

%%
% use the transform function to resize each image
image_size = [98 50];
dsTrain = augmentedImageDatastore(image_size,imdsTrain,'ColorPreprocessing',
'none');
dsVal = augmentedImageDatastore(image_size,imdsVal,'ColorPreprocessing','none');

% define the grid of hyperparameters
num_classes = 12;                                 % number of classes
num_layers = [2, 3, 4];             % number of layers
num_filters = [64, 128];            % number of filters per layer
filter_size=[6,7];                % number of filters

% initialise best solution
best_accuracy = 0;
best_params = [];

% auxiliary parameters
aux_params{1} = num_classes;
aux_params{2} = image_size;

% loop over the grid of hyperparameters
for i = 1:length(num_layers)
    for j = 1:length(num_filters)
        for k = 1:length(filter_size)

% current hyperparameters
hyper_params = [num_layers(i), num_filters(j), filter_size(k)];

% create and train model with current hyperparams
layers = create_model(hyper_params,aux_params);

% train model
options = trainingOptions('adam', ...
    "MiniBatchSize",20, ...
    'InitialLearnRate',0.001, ...
    'MaxEpochs',6, ...
    'Shuffle','every-epoch', ...
    'ValidationData',dsVal, ...
    'ValidationFrequency',10, ...
    'Verbose',true, ...
    'Plots','training-progress',...
    'ExecutionEnvironment','cpu');

% train network
[model,info] = trainNetwork(dsTrain,layers,options);

% extract validation accuracy for current model
accuracy(i,j,k) = info.ValidationAccuracy(end);

% store parameters if they are better than previous
            if accuracy(i,j,k) > best_accuracy
                    best_accuracy = accuracy;
                    best_params = hyper_params;
                end
            end
        end
end

% display best hyperparameters
disp(['Best hyperparameters: ' num2str(best_params)])
%%
% define a function to create a model
function layers = create_model(hyper_params,aux_params)
% unpack hyperparameter values under test
num_layers = hyper_params(1);
num_filters = hyper_params(2);
filter_size = hyper_params(3);
% unpack auxiliary parameters needed to build network
num_classes = aux_params{1};
image_size = aux_params{2};
% create input layer
layers = [
imageInputLayer(image_size)
];
% create blocks of conv -> batch norm -> relu -> max pool layers
for i = 1:num_layers
layers = [layers
convolution2dLayer(filter_size,num_filters,'Padding','same')
batchNormalizationLayer
reluLayer
maxPooling2dLayer(2,'Stride',2)];
end
% output layers
layers = [layers
maxPooling2dLayer(2,'Stride',2)
fullyConnectedLayer(num_classes)
softmaxLayer
classificationLayer];
end
```

Depth -Wise

```
clear

% define the random number seed for repeatable results
rng(1,'twister');

%% Load Speech Data

% create an image data store from the raw images
imdsTrain = imageDatastore('speechImageData\TrainData',...
"IncludeSubfolders",true,"LabelSource","foldernames");

% create an image validation data store from the validation images
imdsVal = imageDatastore('speechImageData\ValData',...
"IncludeSubfolders",true,"LabelSource","foldernames");

%%
% use the transform function to resize each image
```

```
image_size = [98 50];
dsTrain = augmentedImageDatastore(image_size,imdsTrain,'ColorPreprocessing',
'none');
dsVal = augmentedImageDatastore(image_size,imdsVal,'ColorPreprocessing','none');
% define constant parameters
num_classes = 12;  % number of classes
num_filters = 32;  % base number of filters in convolutional layers
filter_size = 7;  % convolutional filter size

% define network layers
layers = [
    imageInputLayer([image_size 1])

    convolution2dLayer(7,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    groupedConvolution2dLayer(3,1,'channel-wise','Stride',2,'Padding','same')
    batchNormalizationLayer
    reluLayer

    convolution2dLayer(7,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    convolution2dLayer(7,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer


    convolution2dLayer(7,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    groupedConvolution2dLayer(3,1,'channel-wise','Stride',2,'Padding','same')
    batchNormalizationLayer
    reluLayer

    convolution2dLayer(7,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer
    dropoutLayer(0.2)

    fullyConnectedLayer(num_classes)
    softmaxLayer
    classificationLayer];
% display network design
analyzeNetwork(layers)
% training options
options = trainingOptions('adam', ...
    "MiniBatchSize",20, ...
    'InitialLearnRate',0.001, ...
    'MaxEpochs',6, ...
    'Shuffle','every-epoch', ...
    'ValidationData',dsVal, ...
    'ValidationFrequency',10, ...
    'Verbose',true, ...
    'Plots','training-progress',...
    'ExecutionEnvironment','cpu');
```

```
% train network
net = trainNetwork(dsTrain,layers,options);
% classify the validation output using the trained network
[YPred,probs] = classify(net,dsVal);
% extract ground truth labels
YVal = imdsVal.Labels;
% accuracy in percent
accuracy = 100*sum(YPred == YVal)/numel(YVal);
disp(['The accuracy is: ' num2str(accuracy)])

% plot confusion matrix
figure;
plotconfusion(YVal,YPred)
% Display sample test images with predicted labels and
% the predicted probabilities of the images having those labels.
idx = randperm(numel(imdsVal.Files),16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(imdsVal,idx(i));
    imshow(I)
    label = YPred(idx(i));
    title(string(label)+ ", "+num2str(100*max(probs(idx(i),:)),3)+"%");
end
disp(["Validation Set Accuracy: " num2str(accuracy) "%"]);
```

Model Averaging

```
clear

% define the random number seed for repeatable results
rng(1,'twister');

%% Load Speech Data

% create an image data store from the raw images
imdsTrain = imageDatastore('speechImageData\TrainData',...
"IncludeSubfolders",true,"LabelSource","foldernames");

% create an image validation data store from the validation images
imdsVal = imageDatastore('speechImageData\ValData',...
"IncludeSubfolders",true,"LabelSource","foldernames");

%%
% use the transform function to resize each image
image_size = [98 50];
dsTrain = augmentedImageDatastore(image_size,imdsTrain,'ColorPreprocessing',
'none');
dsVal = augmentedImageDatastore(image_size,imdsVal,'ColorPreprocessing','none');
%%
%Training Set
shuffled_idx1 = randperm(dsTrain.NumObservations);
shuffled_idx2 = randperm(dsTrain.NumObservations);
shuffled_idx3 = randperm(dsTrain.NumObservations);
subset_size1 = floor(dsTrain.NumObservations);
subset_size2 = floor(dsTrain.NumObservations);
subset_size3 = floor(dsTrain.NumObservations);
```

```
dsTrainsubset1 = subset(dsTrain, shuffled_idx1(1:subset_size1));
dsTrainsubset2 = subset(dsTrain, shuffled_idx2(1:subset_size2));
dsTrainsubset3 = subset(dsTrain, shuffled_idx3(1:subset_size3));

% Validation Set
shuffled_idx4 = randperm(dsVal.NumObservations);
shuffled_idx5 = randperm(dsVal.NumObservations);
shuffled_idx6 = randperm(dsVal.NumObservations);
subset_size4 = floor(dsVal.NumObservations);
subset_size5 = floor(dsVal.NumObservations);
subset_size6 = floor(dsVal.NumObservations);

dsValsubset1 = subset(dsVal, shuffled_idx4(1:subset_size4));
dsValsubset2 = subset(dsVal, shuffled_idx5(1:subset_size5));
dsValsubset3 = subset(dsVal, shuffled_idx6(1:subset_size6));
%% model 1
% define constant parameters
num_classes = 12;  % number of classes
num_filters = 32;  % base number of filters in convolutional layers
filter_size = 7;  % convolutional filter size
%%
% define network layers
layers = [
    imageInputLayer([image_size 1])

    convolution2dLayer(filter_size,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(filter_size,num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(filter_size,2*num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(filter_size,2*num_filters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    dropoutLayer(0.2)

    maxPooling2dLayer([12,1])

    fullyConnectedLayer(num_classes)
    softmaxLayer
    classificationLayer];
%%
% display network design
analyzeNetwork(layers)
% training options
```

```
options = trainingOptions('adam', ...
    "MiniBatchSize",20, ...
    'InitialLearnRate',0.001, ...
    'MaxEpochs',6, ...
    'Shuffle','every-epoch', ...
    'ValidationData',dsVal, ...
    'ValidationFrequency',10, ...
    'Verbose',true, ...
    'Plots','training-progress',...
    'ExecutionEnvironment','cpu');

% train network
net = trainNetwork(dsTrainsubset1,layers,options);
%% model 2
% define constant parameters
num_classes2 = 12;  % number of classes
num_filters2 = 64;  % base number of filters in convolutional layers
filter_size2 = 6;  % convolutional filter size
%%
% define network layers
layers2 = [
    imageInputLayer([image_size 1])

    convolution2dLayer(filter_size2,num_filters2,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(filter_size2,num_filters2,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(filter_size2,2*num_filters2,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(filter_size2,2*num_filters2,'Padding','same')
    batchNormalizationLayer
    reluLayer

    dropoutLayer(0.2)

    maxPooling2dLayer([12,1])

    fullyConnectedLayer(num_classes)
    softmaxLayer
    classificationLayer];
%%
% display network design
analyzeNetwork(layers2)
% training options
options_2 = trainingOptions('adam', ...
    "MiniBatchSize",20, ...
    'InitialLearnRate',0.001, ...
    'MaxEpochs',6, ...
    'Shuffle','every-epoch', ...
    'ValidationData',dsVal, ...
    'ValidationFrequency',10, ...
    'Verbose',true, ...
    'Plots','training-progress',...
    'ExecutionEnvironment','cpu');

% train network
net_2 = trainNetwork(dsTrainsubset2,layers2,options_2);
%% model 3
% define constant parameters
num_classes = 12;  % number of classes
num_filters3 = 32;  % base number of filters in convolutional layers
filter_size3 = 7;  % convolutional filter size

% define network layers
layers3 = [
    imageInputLayer([image_size 1])

    convolution2dLayer(7,num_filters3,'Padding','same')
    batchNormalizationLayer
    reluLayer

    groupedConvolution2dLayer(3,1,'channel-wise','Stride',2,'Padding','same')
    batchNormalizationLayer
    reluLayer

    convolution2dLayer(7,num_filters3,'Padding','same')
    batchNormalizationLayer
    reluLayer

    convolution2dLayer(7,num_filters3,'Padding','same')
    batchNormalizationLayer
    reluLayer


    convolution2dLayer(7,num_filters3,'Padding','same')
    batchNormalizationLayer
    reluLayer

    groupedConvolution2dLayer(3,1,'channel-wise','Stride',2,'Padding','same')
    batchNormalizationLayer
    reluLayer

    convolution2dLayer(7,num_filters3,'Padding','same')
    batchNormalizationLayer
    reluLayer
    dropoutLayer(0.2)

    fullyConnectedLayer(num_classes)
    softmaxLayer
    classificationLayer];
% display network design
analyzeNetwork(layers3)
% training options
options3 = trainingOptions('adam', ...
    "MiniBatchSize",20, ...
    'InitialLearnRate',0.001, ...
    'MaxEpochs',6, ...
    'Shuffle','every-epoch', ...
    'ValidationData',dsVal, ...
    'ValidationFrequency',10, ...
    'Verbose',true, ...
    'Plots','training-progress',...
    'ExecutionEnvironment','cpu');

% train network
net_3 = trainNetwork(dsTrainsubset3,layers3,options3);
%% Predicitions
% classify the validation output using the trained network 1
YPred1= classify(net,dsVal);
% extract ground truth labels
YVal = imdsVal.Labels;

% accuracy in percent
accuracy1 = 100*sum(YPred1 == YVal)/numel(YVal);
% classify the validation output using the trained network
YPred2 = classify(net_2,dsVal);
% extract ground truth labels
YVal = imdsVal.Labels;

% accuracy in percent
accuracy2 = 100*sum(YPred2 == YVal)/numel(YVal);
% classify the validation output using the trained network
YPred3 = classify(net_3,dsVal);
% extract ground truth labels
YVal = imdsVal.Labels;

% accuracy in percent
accuracy3 = 100*sum(YPred3 == YVal)/numel(YVal);

%Ensemble pred
ensemblePred = mode([YPred1, YPred2, YPred3],2);
ensembleaccuracy = 100*sum(ensemblePred == YVal)/numel(YVal);
disp(["Validation Set Accuracy1: " num2str(accuracy1) "%"]);
disp(["Validation Set Accuracy2: " num2str(accuracy2) "%"]);
disp(["Validation Set Accuracy3: " num2str(accuracy3) "%"]);
disp(["Validation Set Ensemble Accuracy: " num2str(ensembleaccuracy) "%"]);
```

Deep Network Designer – Resnet

Create and Train a Deep Learning Model

Script for creating and training a deep learning network with the following properties:

```
Number of layers: 21

Number of connections: 20

Training setup file: D:\Deep Learning\Speech recognition\params_2023_03_27__23_45_11.mat
```

Run this script to create the network layers, import training and validation data, and train the network. The network layers are stored in the workspace variable `layers`. The trained network is stored in the workspace variable `net`.

To learn more, see Generate MATLAB Code From Deep Network Designer.

Auto-generated by MATLAB on 27-Mar-2023 23:45:19

Load Initial Parameters

Load parameters for network initialization. For transfer learning, the network initialization parameters are the parameters of the initial pretrained network.

```
trainingSetup = load("D:\Deep Learning\Speech recognition\params_2023_03_27__23_45_11.mat");
```

Import Data

Import training and validation data.

```
imdsTrain = imageDatastore("D:\Deep Learning\Speech recognition\speechImageData\TrainData","IncludeSubfolders",true,"LabelSource","foldernames");

imdsValidation = imageDatastore("D:\Deep Learning\Speech recognition\speechImageData\ValData","IncludeSubfolders",true,"LabelSource","foldernames");



% Resize the images to match the network input layer.
augimdsTrain = augmentedImageDatastore([98 50 1],imdsTrain);

augimdsValidation = augmentedImageDatastore([98 50 1],imdsValidation);
```

Set Training Options

Specify options to use when training.

```
opts = trainingOptions("adam",...

    "ExecutionEnvironment","cpu",...

    "InitialLearnRate",0.001,...

    "MaxEpochs",6,...

    "MiniBatchSize",20,...

    "Shuffle","every-epoch",...

    "ValidationFrequency",10,...

    "Plots","training-progress",...

    "ValidationData",augimdsValidation);
```

Create Array of Layers

```
layers = [
    imageInputLayer([98 50 1],"Name","imageinput")
    convolution2dLayer([7 7],128,"Name","conv","Padding","same","Stride",[2
2])
    batchNormalizationLayer("Name","batchnorm")
    reluLayer("Name","relu")
    maxPooling2dLayer([2 2],"Name","maxpool","Padding","same","Stride",[2 2])
    convolution2dLayer([7 7],128,"Name","conv_1","Padding","same","Stride",[2
2])
    batchNormalizationLayer("Name","batchnorm_1")
    reluLayer("Name","relu_1")
    maxPooling2dLayer([2 2],"Name","maxpool_1","Padding","same","Stride",[2
2])
    convolution2dLayer([7 7],128,"Name","conv_2","Padding","same","Stride",[2
2])
    batchNormalizationLayer("Name","batchnorm_3")
    reluLayer("Name","relu_3")
    maxPooling2dLayer([2 2],"Name","maxpool_2","Padding","same","Stride",[2
2])
    convolution2dLayer([7 7],128,"Name","conv_3","Padding","same","Stride",[2
2])
    batchNormalizationLayer("Name","batchnorm_2")
    reluLayer("Name","relu_2")
    dropoutLayer(0.2,"Name","dropout")
    maxPooling2dLayer([12 12],"Name","maxpool_3","Padding","same")
    fullyConnectedLayer(12,"Name","fc")
    softmaxLayer("Name","softmax")
    classificationLayer("Name","classoutput")];
```

Train Network

Train the network using the specified options and training data.

```
[net, traininfo] = trainNetwork(augimdsTrain,layers,opts);
```