Mitigate Communication Barrier!

Recognise Sign Language using Neural Networks and Deep Learning

Presented by: Group 21
Jialin Liu (A0172938R)
Mansi Agarwal (A0218968J)
Nishtha Malhotra (A0228556W)

CS5242 Neural Networks and Deep Learning
AY 2022/2023

Contents

1. Introduction

- ❖ Project Motivation
- ❖ Problem Statement
- ❖ Challenges

Problem | Preparation | Model | Selection | Conclusion

Widespread Hearing Loss

Nearly **430 million** people suffer from deafness or hearing loss *[Source: WHO]*

Singapore Association for the Deaf (**SADeaf**) estimates **500,000** people suffer with hearing loss, only 1800 know sign language

THE SINGAPORE ASSOCIATION FOR THE DEAF
SIGN LANGUAGE INTERPRETATION

Automatic Sign Language Recognition

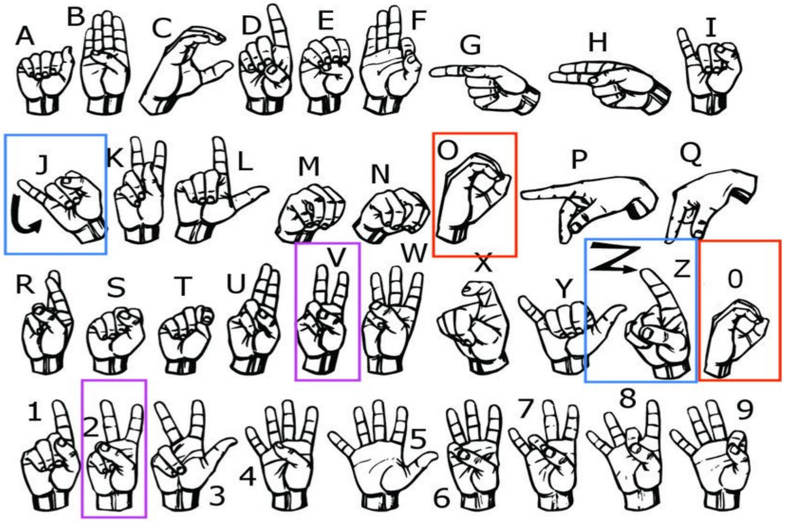
Significance

- **Mitigate communication barrier** for hard-of-hearing community with their First Responders and abled communities
- Create Educational resources, Work opportunities, Foster friendly professional environments
- Build easily accessible **sign language instruction** and specialized education programs

Automatic Sign Language Challenges

Challenges

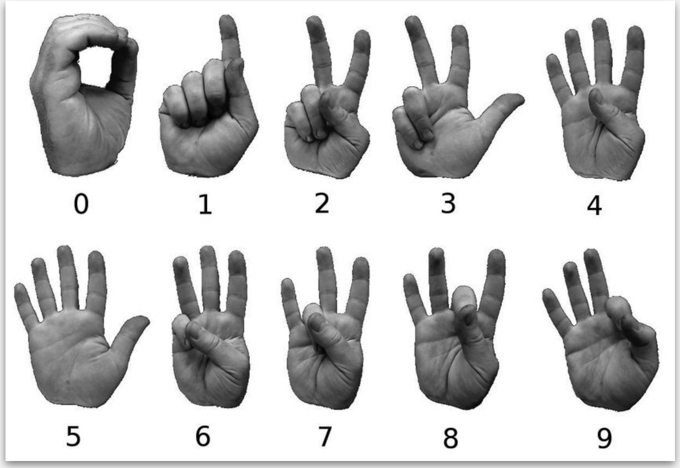
- Multimodal: Signs are characterized by hand shapes, movements, facial expressions, body posters
- Similar Signs for different characters
- Variations due to orientation, hand shape of different people
- Characters with Dynamic Hand Gestures cannot be recognised by static image recognition systems



★ Similar signs with clenched fist for alphabets A, E, M, N, S, T
★ Similar signs for characters pairs ‘2’ | ‘V’ and ‘O’ | ‘0’
★ Dynamic hand gestures characteristic to alphabets ‘J’ and ‘Z’

Problem Statement

Build Supervised Learning
American Sign Language (ASL)
Recogniser System
using deep neural networks
capable of classifying static hand gesture
images into corresponding 10 digit labels



2. Data

- Collection
- Cleaning
- Exploration

Problem | Preparation | Model | Selection | Conclusion

Data Collection

Image Scrapers

Istockphoto and gettyimages

- Beautiful Soup: Obtain all image links using unique html element containing the image
- Different URLs can be provided to scrapper to get images from URL link

Google

- Used Google API ***images-scraper*** for collecting digit and character images from Google Images

Scrapped Images Statistics		
Istockphoto	GettyImages	Google
5000	5000	1440

3 Sources

iStockphoto

gettyimages

Google

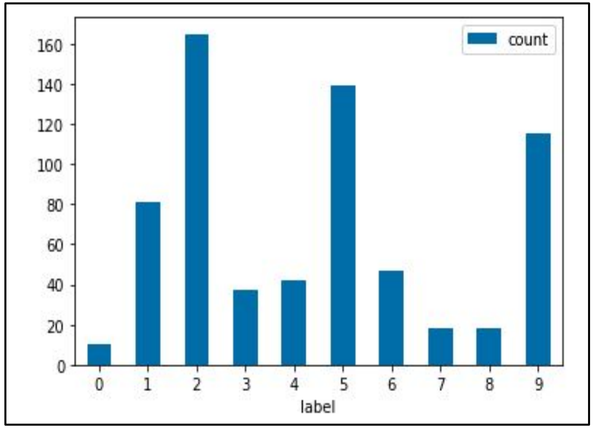
Data Cleaning and Exploration

Cleaning

- Removed noisy images irrelevant to our use case of digit recognition

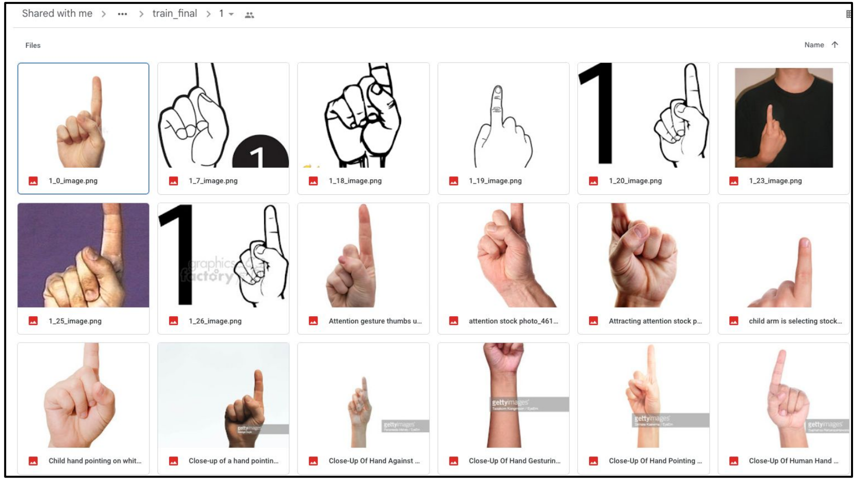
Exploration

- Colour plays no role - Single Channel input images
- Challenges and Problems in collected images
    - Features difference in animated and human hand images
    - Imbalance sample size for 10 digits Label distribution
    - Needs standardisation of size and resolution



Distribution of sample size by digits as labels

Training and Test Dataset



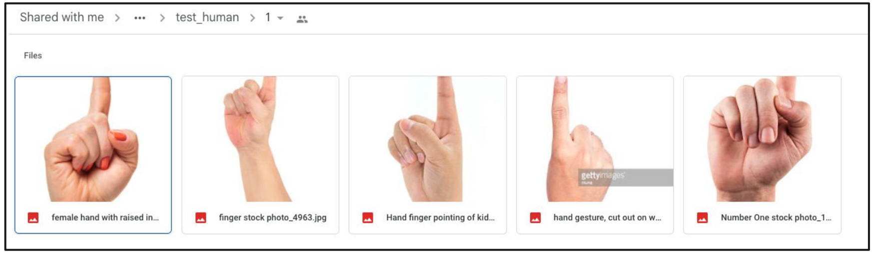
<u>**Randomly split cleaned data into train and test**</u>

Training dataset

- 672 Images
- Both human and animated hands

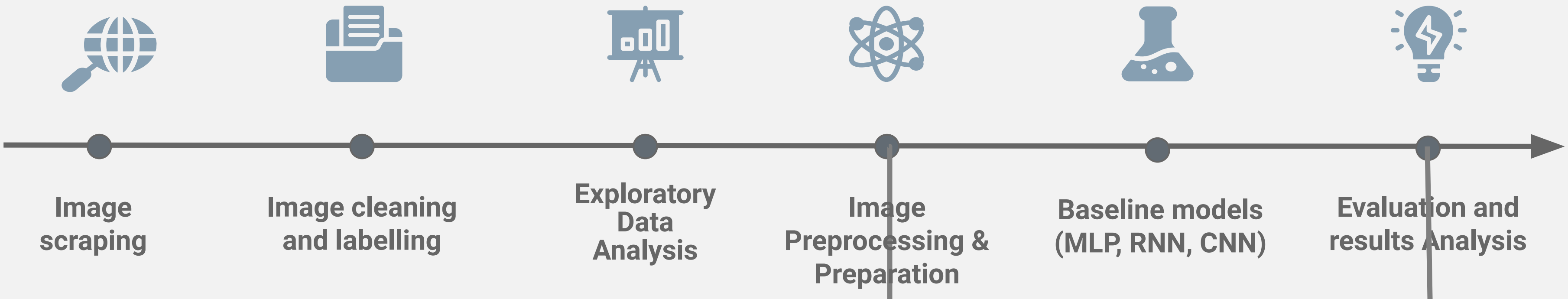
Test dataset

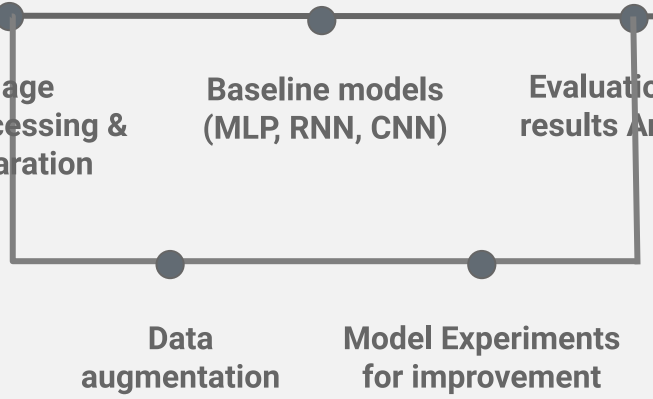
- 50 Images | 5 for each digit
- Only human hands more relevant to real-world use case



3. Deep Learning Models Training and Evaluation

Methodology





Standardised following considerations in all models

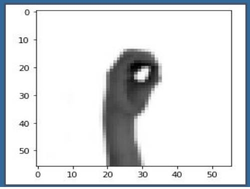
- **Setting random seed:** for reproducibility
- **Optimizer:** Adaptive learning rate algorithms like Adam.
- **Train batch size**: 16
- **Model hyperparameters:** learning rate, #epochs, #hidden layers, #hidden units

Image Preprocessing and Preparation

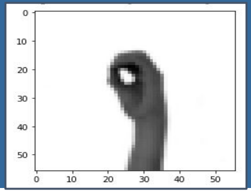
Image Transformation

- **Re-size:** 28x28, 56x56, 112x112, 224x224
- **Cropping from centre**
  - Given size as input, image cropped from centre
  - Automatically padded on edges for given size
- **Gray Scale:** Reduces to single channel
- **Feature Scaling and Normalisation:**
  - Converts PIL image to numpy array of range of [0,1]
  - Normalisation rescales images to have a mean of 0.485 and a standard deviation of 0.224

Original



Horizontal Flip



Data Augmentation

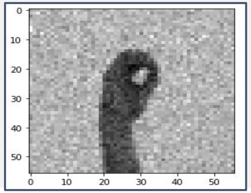
Advantages:

- Incorporates invariances and generalization
- Provides more samples for training

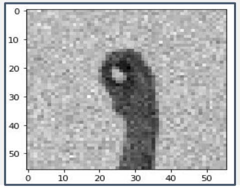
Simulated real-world variances:

- Original Image transformations
- Horizontal Flip
- Original with Gaussian Noise
- Flipped with Gaussian Noise
- Original with Random Noise

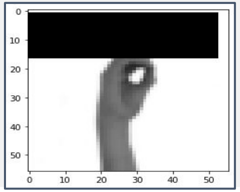
Original with gaussian noise



Flipped with gaussian noise



Original with random erasing



Multilayer Perceptron – *Model setup*

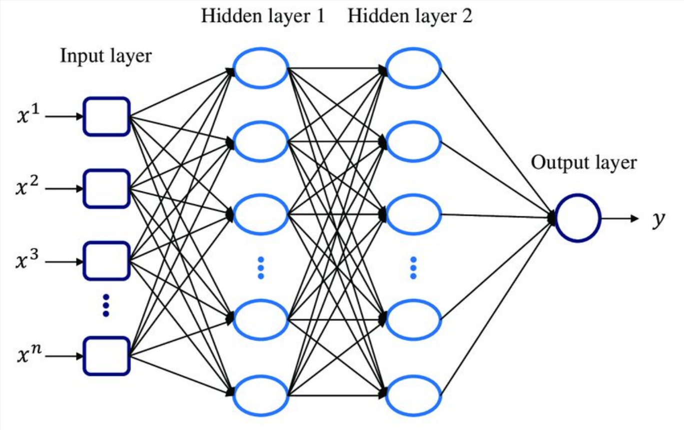
Why MLP?

- Basic Deep Neural Network
- Consists series of fully connected layers

Baseline model

- Input image normalisation
- 2 hidden layers with 120 and 84 neurons each
- Softmax output (10 classes)
- Relu activation function
- 40 epochs
- Adam optimizer

Baseline MLP Architecture




```
MLP(
  (h1): Linear(in_features=784, out_features=120, bias=True)
  (h2): Linear(in_features=120, out_features=84, bias=True)
  (bn1): BatchNorm1d(84, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (out): Linear(in_features=84, out_features=10, bias=True)
)
```

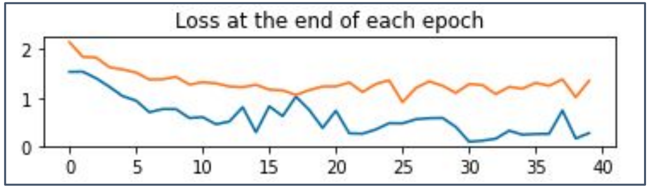
Multilayer Perceptron – *Experiments*

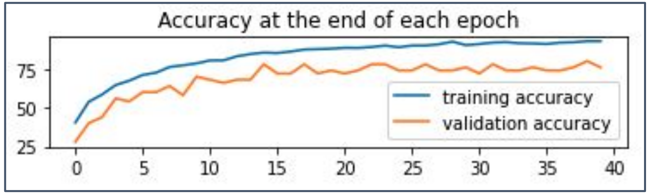
Metric	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5	Experiment 6	Experiment 7
Scenario	28x28 grayscale lr=0.001 Batch Size 16	28x28 grayscale With Data augmentation and Normalization Batch Size 16	56x56 grayscale With Data augmentation and normalisation Batch Size 16	28x28 grayscale With Data augmentation and Normalization Batch Size 16 higher/lower **learning rate**	28x28 grayscale With Data augmentation and Normalization Batch Size 16 more/less **neurons**	28x28 grayscale With Data augmentation and Normalization Batch Size 16 With **He initialisation**	28x28 grayscale With Data augmentation and **BatchNorm** and **batch size 32**
Validation Loss	1.75	0.7	0.3	1.9	0.9 / 0.5	0.5	0.2
Validation Accuracy	16%	52%	40%	10%	34% / 38%	48%	76%

Multilayer Perceptron

Best MLP Model Performance

- **Augmented Training dataset size:** 3360
- **Image size:** 28x28
- **Learning rate**: 0.001
- **Batch size**: 32
- **Num Epochs**: 40
- **Num Neurons**: [120, 84]
- **Batch Normalisation before output layer**





Multilayer Perceptron – *Limitations*

- Spatial information is lost when the image is flattened (matrix to vector) into an MLP
- Would not generalise well if the hand is at different part of the image as correlation of the image features (pixels) is not captured
- It includes too many parameters because it is fully connected. Hence more prone to overfitting
- The very dense web is formed can also resulting in redundancy and inefficiency.

Convolutional Neural Network (CNN)

Why CNN?

- Capable of interpreting spatial relationships between nearby pixels
- Accounts for image stationarity, applies same filter to different parts
- For detailed photos, CNN performs better than a MLP

Baseline CNN Model Setup

- Original Image transformations
- Changed size to 112x112 for better resolution
- Train_batch_size = 16
- # epochs = 20, 50
- Learning_rate = 0.0001

Baseline CNN Model Performance

- Training starts to converge after 30 epochs
- Validation Accuracy range 78% - 84%

Architecture

- ReLU activation function for Conv2D and FC1 layers
- Dropout Layers with p = 0.5

```
CNN(
  (conv1): Conv2d(1, 16, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(16, 16, kernel_size=(5, 5), stride=(1, 1))
  (conv3): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=18432, out_features=256, bias=True)
  (fc2): Linear(in_features=256, out_features=10, bias=True)
)
```

Baseline CNN Model Performance

Metric	Experiment 1 # epochs = 20	Experiment 2 # epochs = 50
Validation Loss	1.0989	1.3452
Validation Accuracy	70%	82%
Test Accuracy	70%	82%

Convolutional Neural Network (CNN)

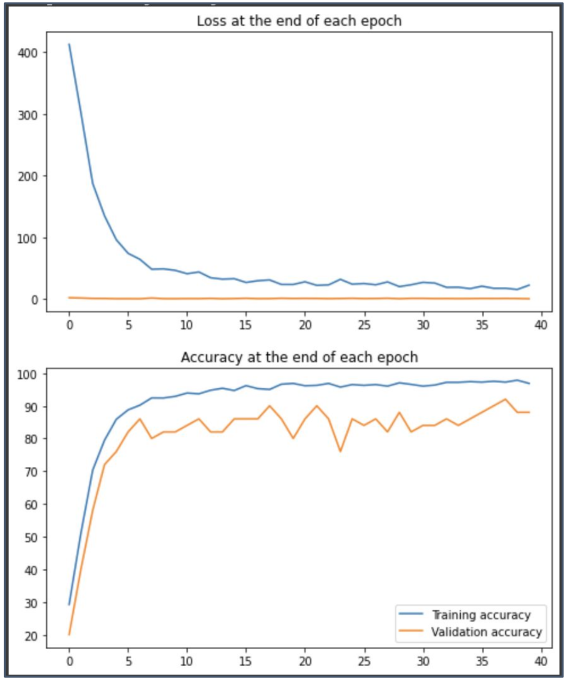
CNN Augmented Data Model Setup

- 5 permutations to original image transformations
- Training data size = 3360 (672*5 times)
- # epochs = 40 | learning_rate = 0.0001 | batch_size = 16
- Trained on same CNN Architecture

CNN Model Performance with Augmented Data

- Training starts to converge after 7 epochs
- Validation Accuracy range 84% - 92%
- Performs better and converged faster than baseline CNN model

CNN Model Performance with Augmented Data		
Metric	**Experiment 1 # epochs = 7**	**Experiment 2 # epochs = 50**
Validation Loss	0.5109	0.5358
Validation Accuracy	86%	92%
Test Accuracy	78%	88%



AlexNet CNN Architecture

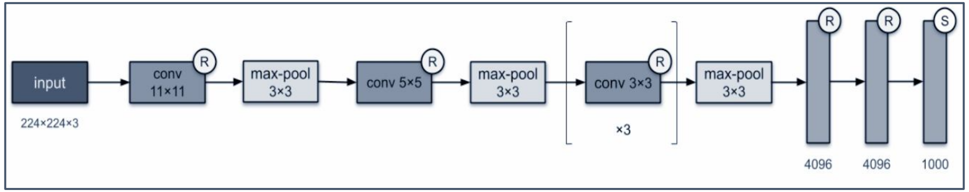
Why AlexNet Architecture?

- Deeper architecture with 8 layers: 5 Conv2D, 3 fully connected
- Overlapping pooling generally find it harder to overfit

AlexNet Baseline Model Setup

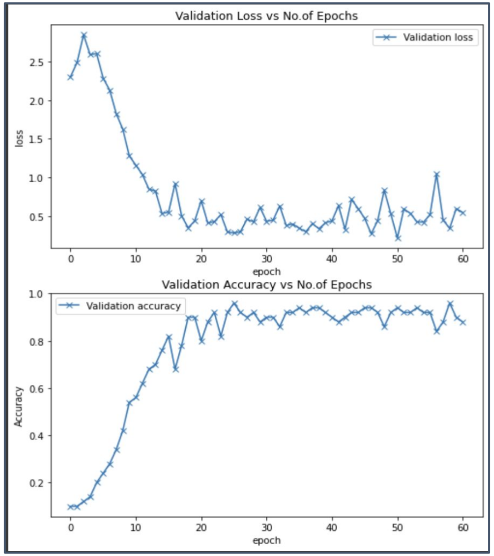
- Resize = 224 x 224 | Input channels = 3
- # epochs = 60 | learning_rate = 0.0001 | batch_size = 16

AlexNet Architecture



AlexNet Baseline Model Performance

- Training starts to converge at 30 epochs
- Validation Accuracy range 86% - 92%



AlexNet CNN Architecture

AlexNet Augmented Data Model Setup

- 5 transformations on original image transformations
- Training data size = 3360 (672*5 times)
- # epochs = 40 | learning_rate = 0.0001 | batch_size = 16
- Addresses the problem of overfitting 60M parameters

AlexNet Augmented Data Model Performance

- Training starts to converge within 10 epochs
- Validation Accuracy range 88% - 94%

Best CNN Model - AlexNet on Augmented Data		
Metric	**Experiment 1 # training_data = 672**	**Experiment 2 # augmented_data = 3360**
Validation Loss	0.3382	0.1173
Validation Accuracy	94% (34th epoch)	96% (epochs = 19th epoch)
Test Accuracy	**92%**	**94%**

Recurrent Neural Network (RNN)

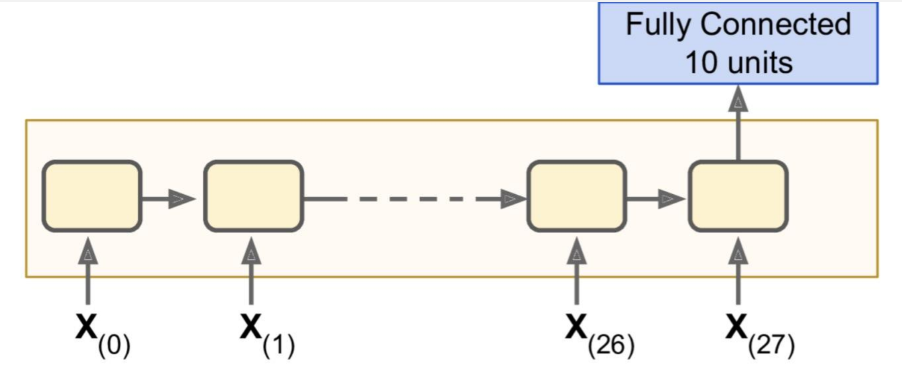
Why RNN?

- Each image is treated as a sequence of 28 rows of 28 pixels (if the image size is 28). Hence RNN is used to study the sign language patterns in these sequences.

Baseline model

- Input image normalisation
- 1 hidden layer RNN with 150 neurons
- 1 FC layer with output size of 10
- 30 epochs
- Adam optimizer

Architecture



Recurrent Neural Network (RNN)

Metric	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5	Experiment 6	Experiment 7
Scenario	28x28 grayscale lr=0.001	28x28 grayscale With Data augmentation Normalization Batch Size = 16	56x56 grayscale With Data augmentation Normalization	112x112 grayscale With Data augmentation Normalization	56x56 grayscale With Data augmentation With more neurons 250 Batch size = 32	28x28 grayscale With Data augmentation Batch Size = 32 Normalization Neurons = 150	28x28 grayscale With Data augmentation Batch Size = 32 Normalization Neurons = 250
Validation Loss	2.8	1.2	1.8	2.0	2.25	1.2	1.3
Validation Accuracy	21%	72%	42%	30%	22%	72%	66%

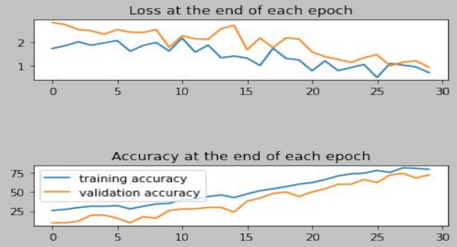
Recurrent Neural Network (RNN)

Image size 28x28

Without Augmentation

Doubled Batch Size

With Augmentation and Norm



Doubled Neurons

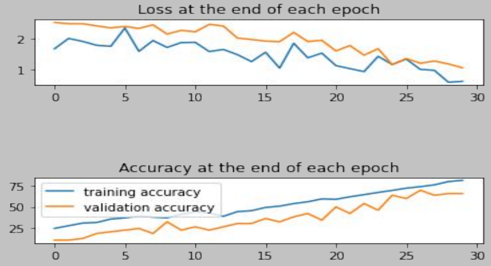
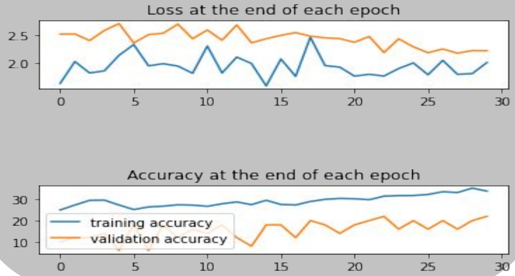


Image size 56x56

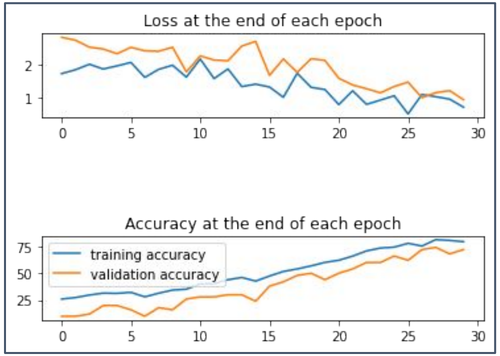
With Augmentation and Norm

Doubled Batch Size & Neurons



Recurrent Neural Network (RNN)

- **Training dataset size (after data augmentation): 3360**
- **Image size: 28x28**
- **Lr: 0.001**
- **Batch size: 16**
- **Num Epochs: 30**
- **Num Neurons: 150**
- **Batch Normalisation before output layer**



36 correct predictions

Conclusion

Deep Neural Network Models Performance Summary			
Model / Metric	**AlexNet CNN** Augmented Training dataset size: 3360 Image size: 3x224x224 Learning Rate: 0.0001 Batch Size: 16 Number of Epochs: 40 5 convolutional layers 3 fully connected layers Dropout, Adaptive Average Pooling	**Multilayer Perceptron MLP** Augmented Training dataset size: 3360 Image size: 28x28 Learning Rate: 0.001 Batch size: 32 Num Epochs: 40 Num Neurons: [120, 84] Batch Normalisation before output layer.	**Recurrent NN RNN** Training dataset size (after data augmentation): 3360 Image size: 28x28 Learning Rate: 0.001 Batch size: 16 Num Epochs: 30 Num Neurons: [150] Batch Normalisation before output layer
Validation Loss	0.1173	0.2	1.2
Validation Accuracy	96% (epochs = 19)	76% (epochs = 40)	72% (epochs = 30)

Project initiatives

- 3 image scrapers for 3 different data sources
  - Can autogroup images into label folders
  - URL/Search based query
- Analyse Epoch vs Validation Loss/Accuracy at every epoch
- Data Augmentation with 5 transformations
- Consolidated images to easily reflect correctly/wrongly labelled test data

Future Developments

- Combine CNN-RNN
- Object Detection to detect the hand and signal
- Model to detect both the animated hands and human hands

Thank You


NUS
National University
of Singapore
National University of Singapore