
# American sign language understanding

Project 1 - TAA

Duarte Mortágua - 92963
Mário Silva - 93430

Pétia Georgieva

# TABLE OF CONTENTS

# 01

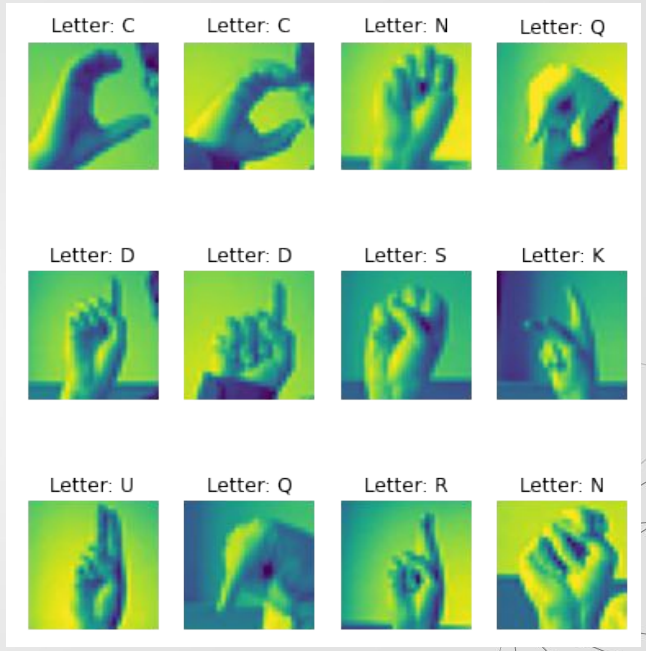
# DATA VISUALIZATION

Why this dataset and data description

# DATA VISUALIZATION

**Training dataset**

| | label | pixel1 | pixel2 | ... | pixel782 | pixel783 | pixel784 |
|---|---|---|---|---|---|---|---|
| 0 | 3 | 107 | 118 | ... | 204 | 203 | 202 |
| 1 | 6 | 155 | 157 | ... | 103 | 135 | 149 |
| 2 | 2 | 187 | 188 | ... | 195 | 194 | 195 |
| 3 | 2 | 211 | 211 | ... | 222 | 229 | 163 |
| 4 | 13 | 164 | 167 | ... | 163 | 164 | 179 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 27450 | 13 | 189 | 189 | ... | 200 | 222 | 225 |
| 27451 | 23 | 151 | 154 | ... | 195 | 195 | 194 |
| 27452 | 18 | 174 | 174 | ... | 202 | 200 | 200 |
| 27453 | 17 | 177 | 181 | ... | 64 | 87 | 93 |
| 27454 | 23 | 179 | 180 | ... | 205 | 209 | 215 |

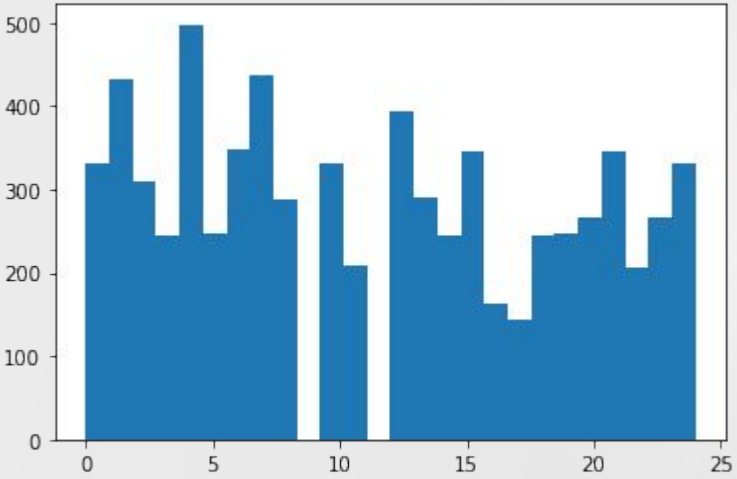



**The author transformed the original dataset:**
cropping to hands-only ⟶ gray-scaling ⟶ resizing

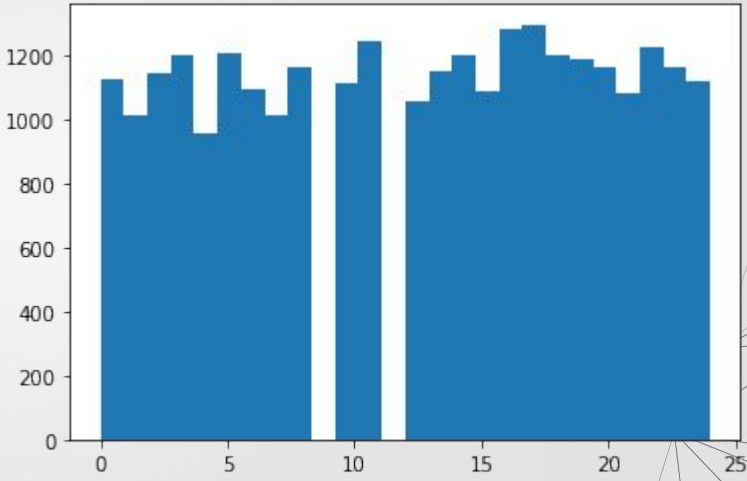
Also creating at least 50+ variations to enlarge the quantity:
Filters ('Mitchell', 'Robidoux', 'Catrom', 'Spline', 'Hermite') ⟶ 5% random pixelation ⟶ +/- 15% brightness/contrast ⟶ 3 degrees rotation.

# DATA VISUALIZATION

**Number of examples/class in the Test Dataset**




**Number of examples/class in the Train Dataset**

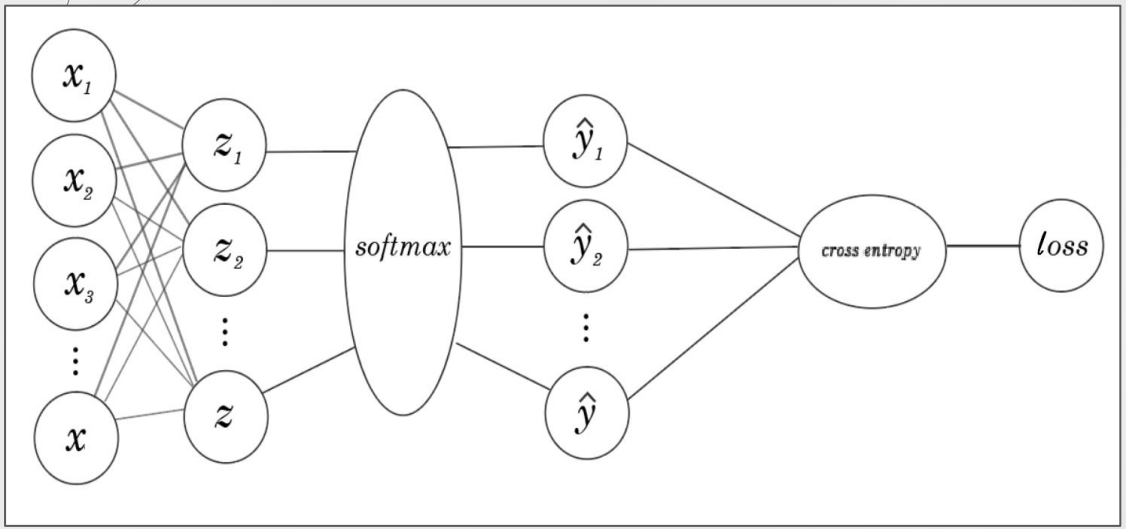


# 02

# ML MODELS

Logistic Regression and
Convolutional Neural Network

# ML MODELS
## LOGISTIC REGRESSION



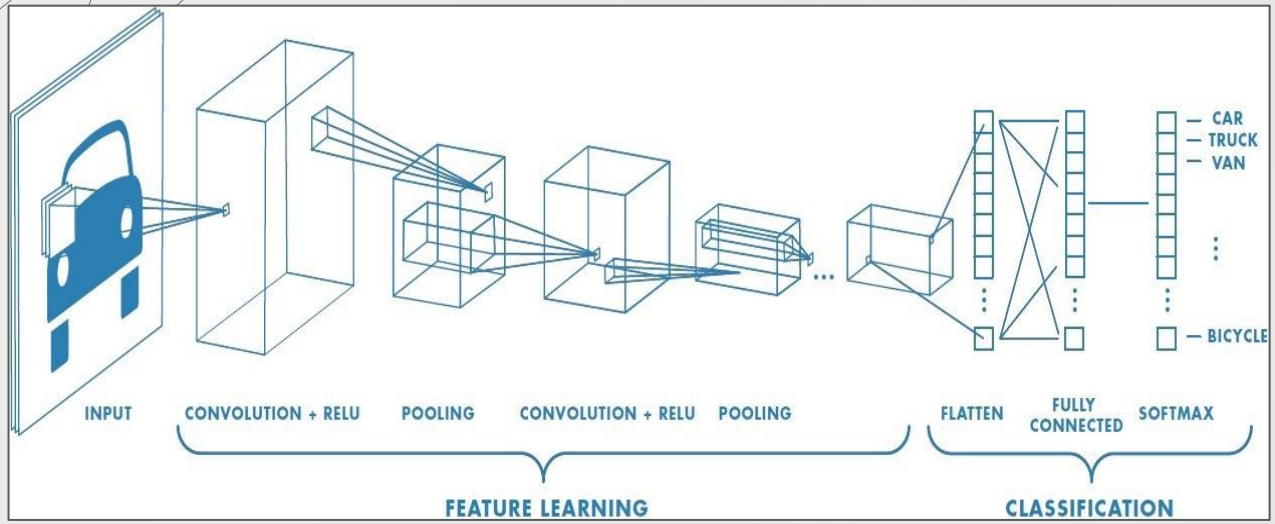

PyTorch

**Linear model with multi class approach**

1. Initially, the function returns a tensor with 26 elements with values ranging from negative infinity to positive infinity

2. cross_entropy function that combines the negative log likelihood and softmax function to normalize the resulting values from the linear function.

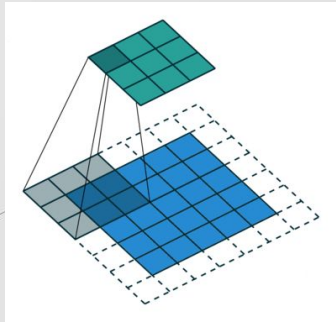
# ML MODELS
## CONVOLUTIONAL NEURAL NETWORK




TensorFlow

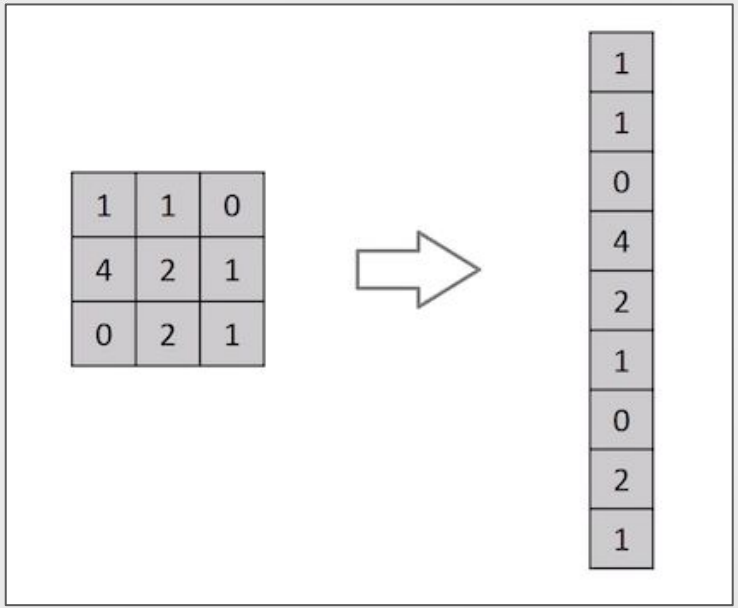
**Deep Learning algorithm**

A ConvNet is able to successfully capture the **Spatial and Temporal dependencies** in an image..

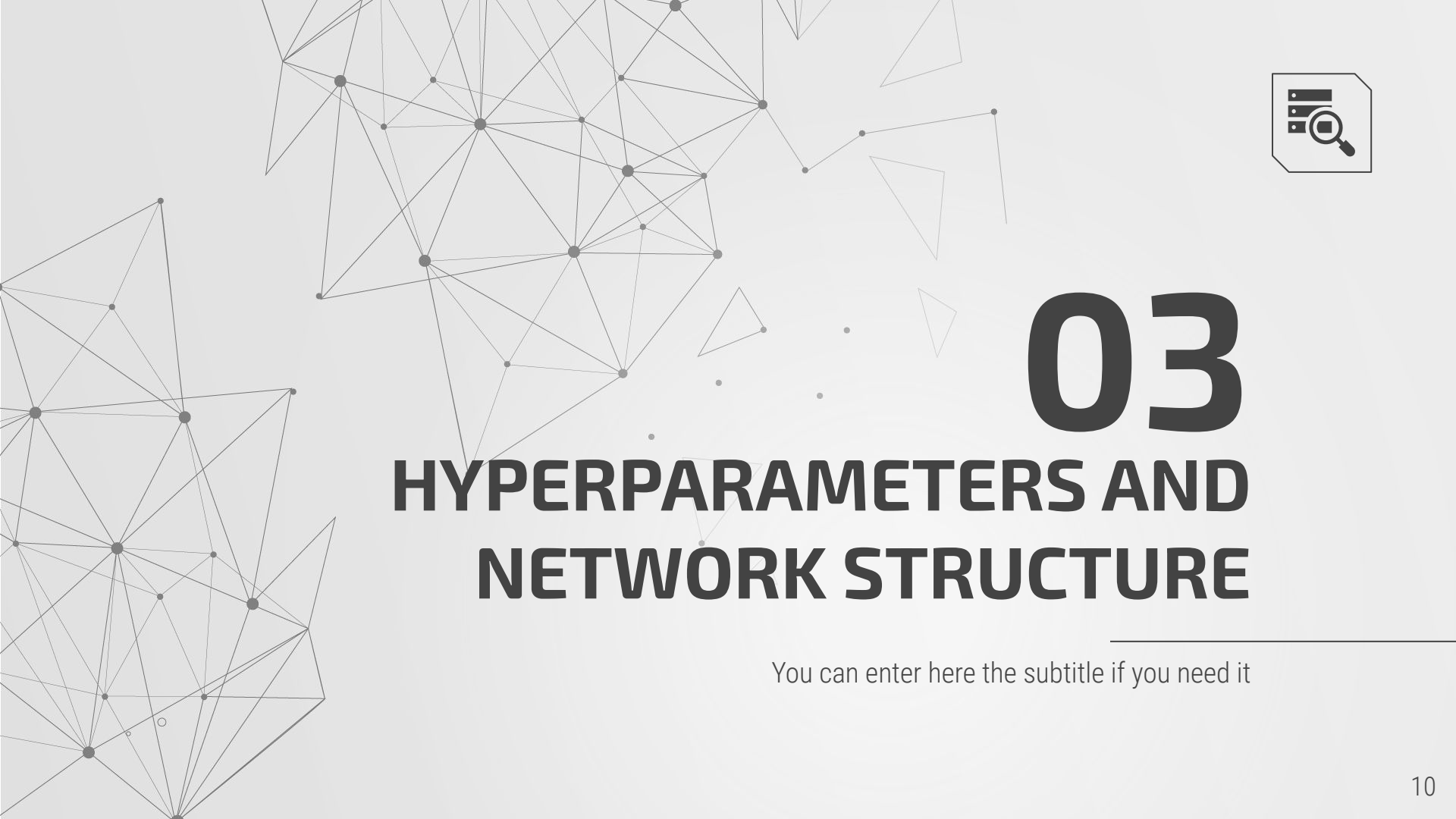


The network can be trained to understand the sophistication of the image better.

# ML MODELS
## WHY 2 DIFFERENT ARCHITECTURES?



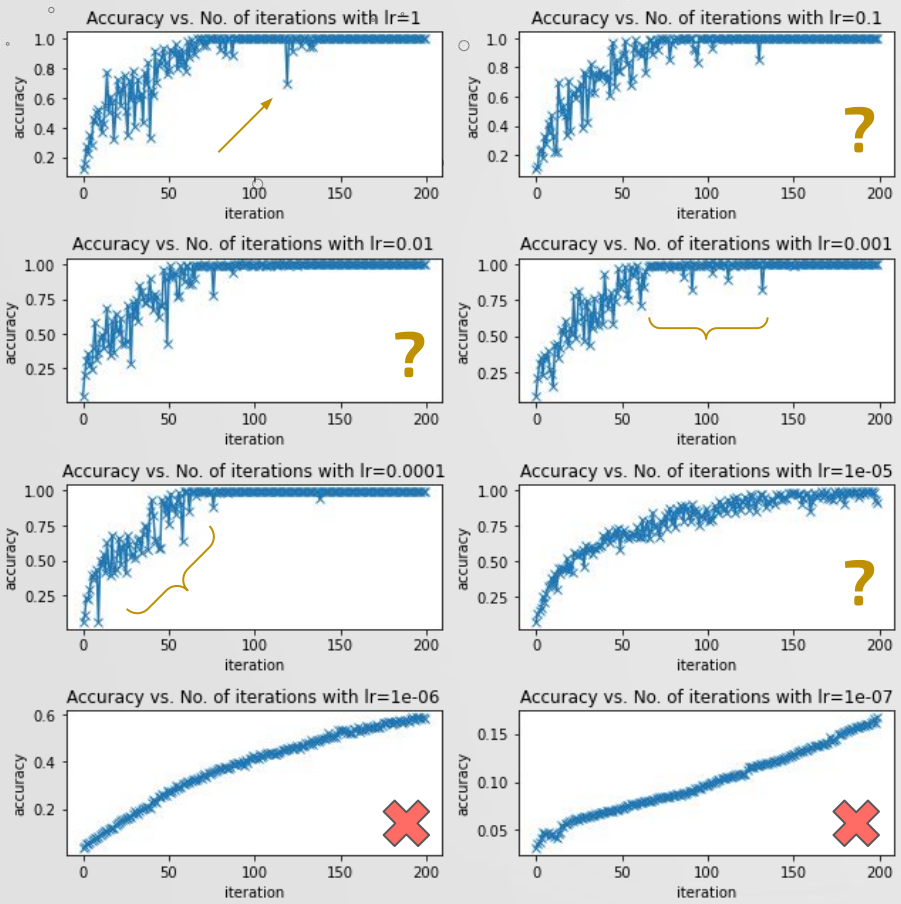

In **cases of extremely basic binary images**, a linear method might show an average precision score while performing prediction of classes, but would have **little to no accuracy when it comes to complex images having pixel dependencies throughout**.



# 03

# HYPERPARAMETERS AND NETWORK STRUCTURE

You can enter here the subtitle if you need it




# HYPERPARAMETERS
## LOGISTIC REGRESSION MODEL

**Batch size and epochs**

256 and 200, based on the literature and prior works related to this problem.

**Learning rate**

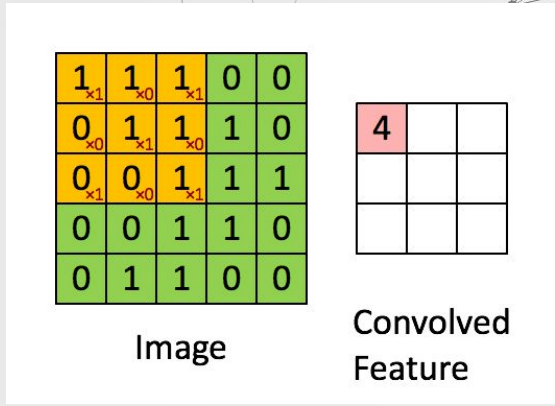
We tested 8 different learning rates and observed the accuracy graphs.

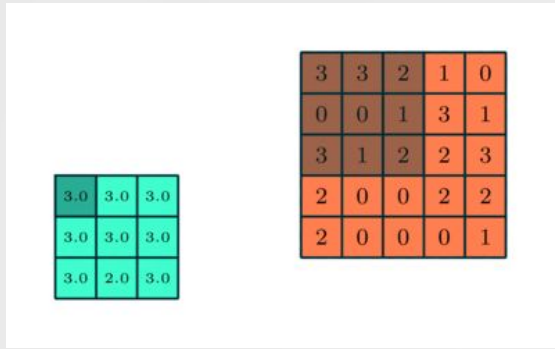
| | | Loss | Accuracy |
|---|---|---|---|
| lr=0.001 **worse** | Validation | 0.4703 | 0.9985 |
| | Test | 1150.3939 | 0.6855 |
| lr=1e-05 **better** | Validation | 0.0703 | 0.9835 |
| | Test | 5.5739 | 0.5628 |

**Convolutional may be better?**

# NETWORK STRUCTURE
## CONVOLUTIONAL NEURAL NETWORK MODEL

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 26, 26, 64)        640
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 11, 11, 128)       73856
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 5, 5, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 3200)              0
_________________________________________________________________
dense_4 (Dense)              (None, 256)               819456
_________________________________________________________________
dense_5 (Dense)              (None, 26)                6682
=================================================================
```



# Data Augmentation

Re-scales pixels of the image to 0-1 by dividing by 255.

Rotates images between 0 and 45 degrees.

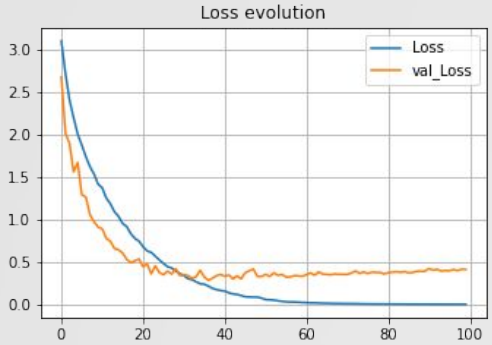
Shifts images horizontally and vertically by 15%.

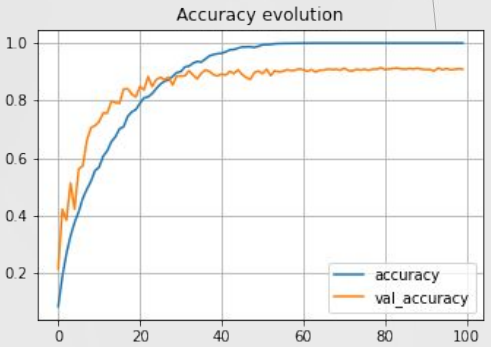
Zooms in and out images by 20%.

Flips images horizontally.

# CNN Hyperparameters

## EPOCHS




## Batch Size

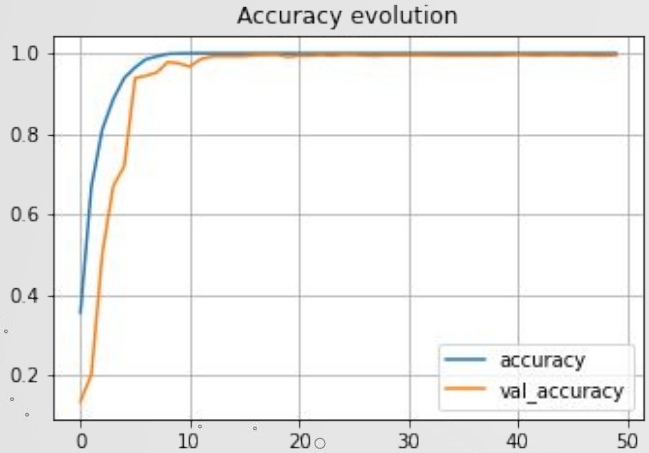
| *Batch Sizes* | *Test Accuracies* |
|---|---|
| **32** | 91% |
| **64** | 95% |
| **128** | 93% |
| **256** | 92% |
| **512** | 98% |
| **1024** | 93% |

## Decaying Learning Rate

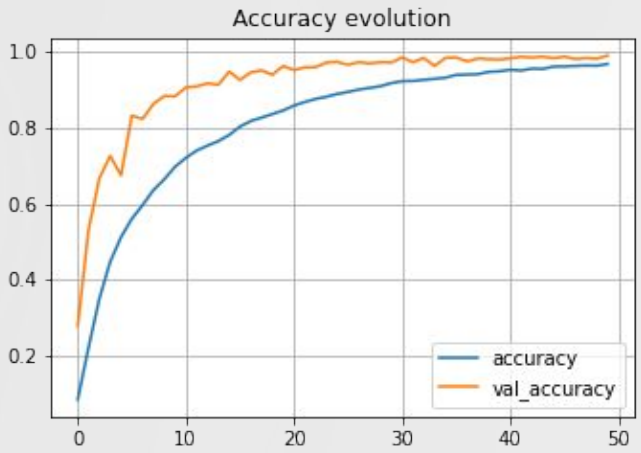
If the validation accuracies were fluctuating a lot the model could overshoot the optima. However, in our case, it didn't make much difference.

# Changes in the Neural Network Structure

## Batch Normalization Layer




## Dropout Layer



# 04

# RESULTS AND PERFORMANCE COMPARISON

Accuracy and Loss.
Comparison between models.

# Accuracies and Losses

## Logistic Regression Model

| | Loss | Accuracy |
|---|---|---|
| Validation | 0.0703 | 0.9835 |
| Test | 5.5739 | 0.5628 |

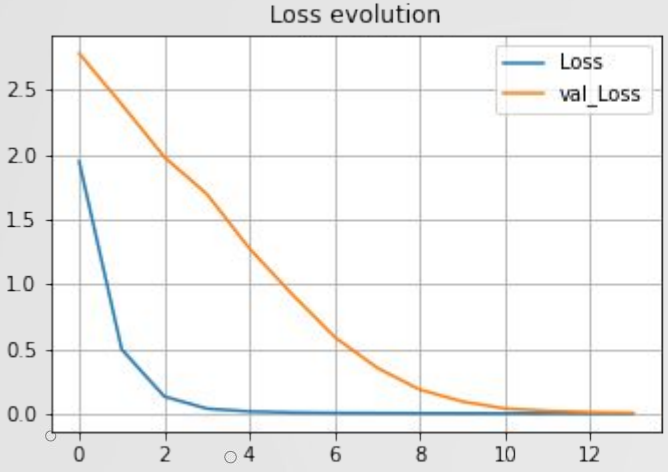
## CNN Model Without Data Augmentation

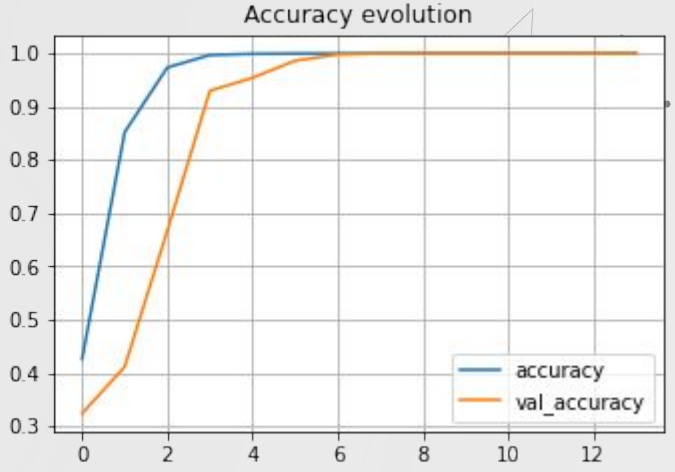
| | Loss | Accuracy |
|---|---|---|
| Validation | 0.0044 | 1.0 |
| Test | 0.2359 | 0.9331 |

## CNN Model With Data Augmentation

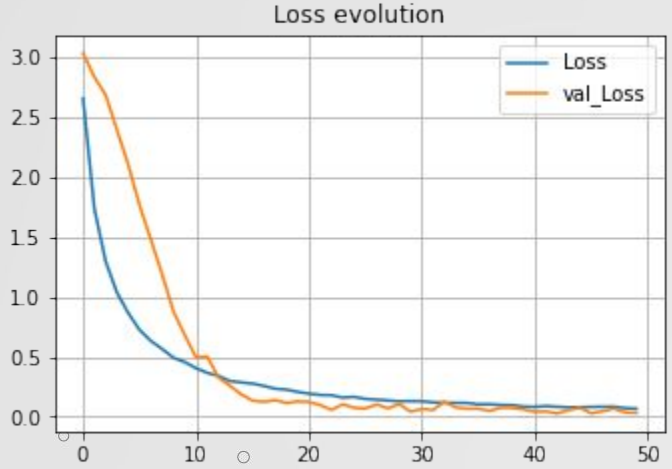
| | Loss | Accuracy |
|---|---|---|
| Validation | 0.0314 | 0.9950 |
| Test | 0.0315 | 0.9873 |

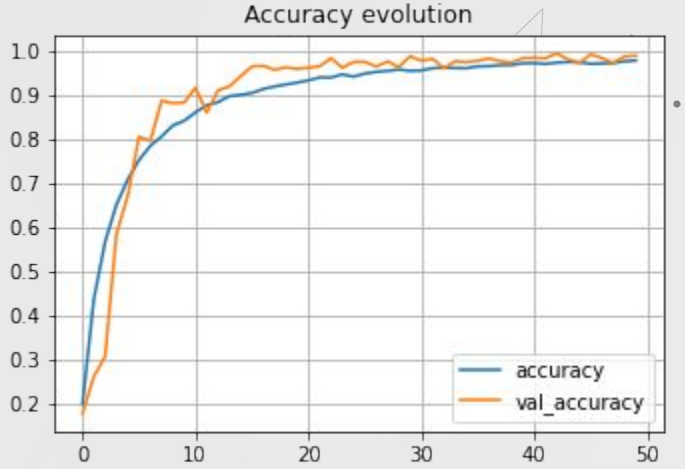
# CNN Model Without Data Augmentation



# CNN Model With Data Augmentation



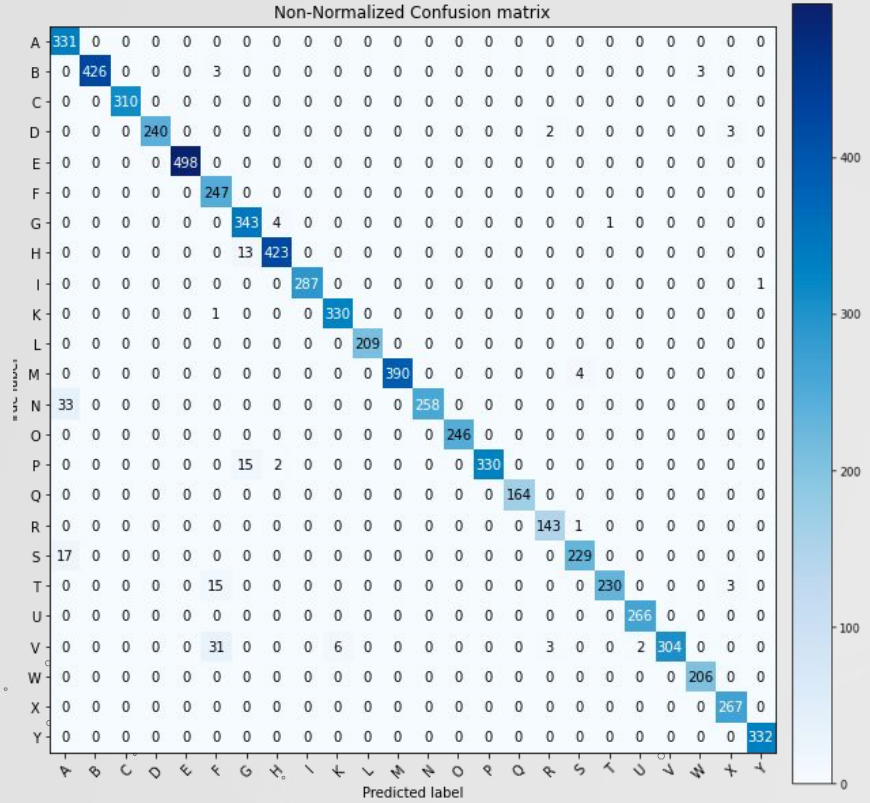



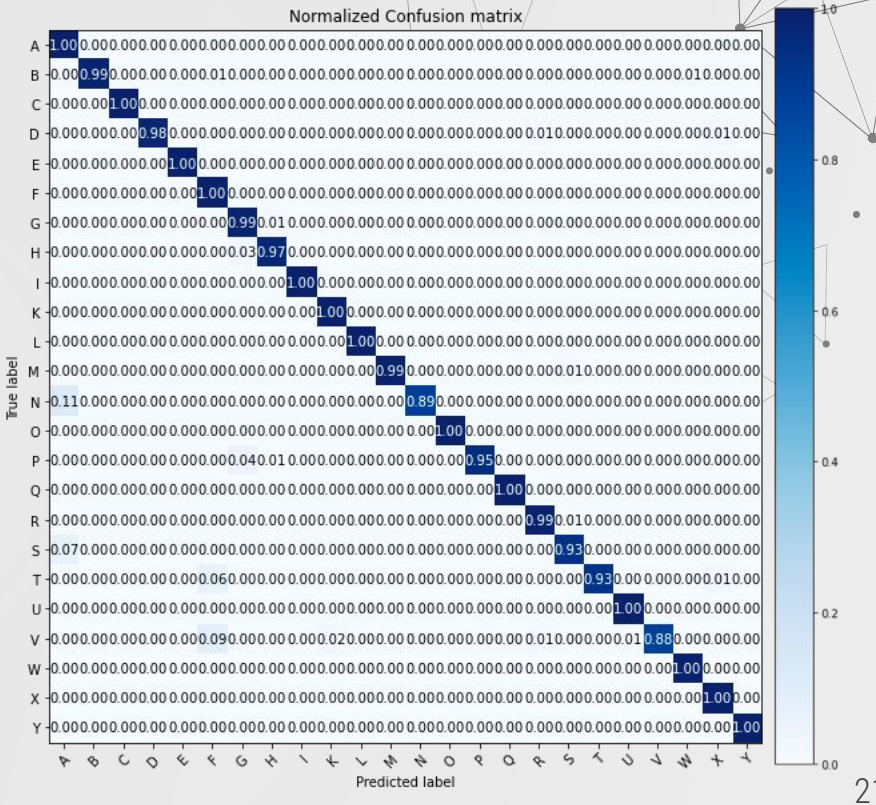
# 05

# PREDICTIONS WITH DIFFERENT PICTURES

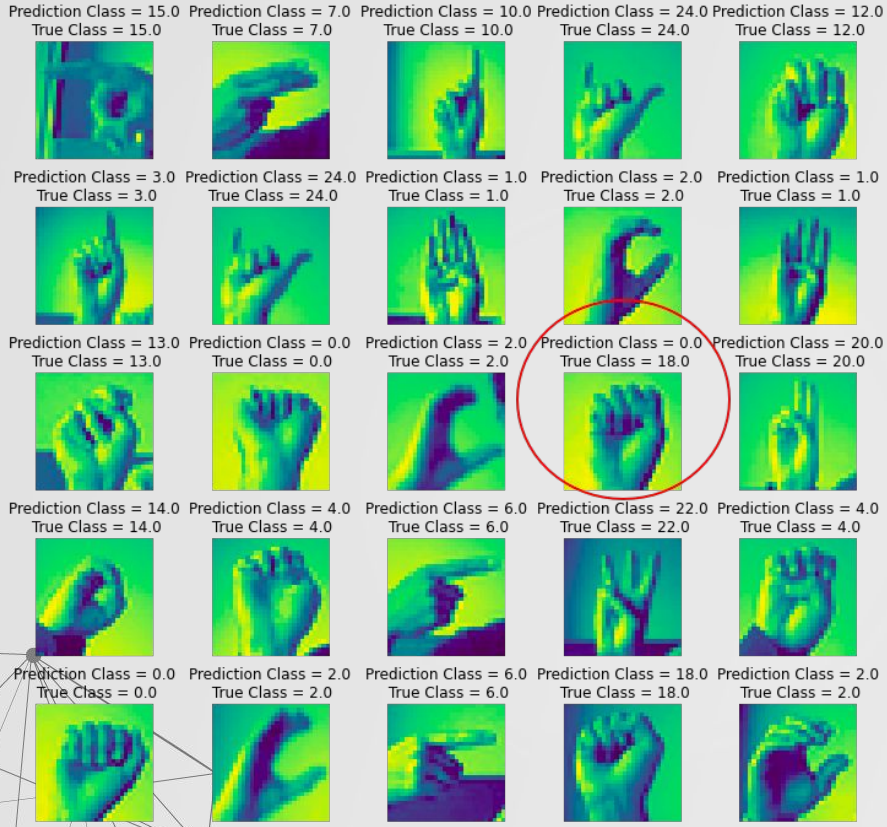
Predicting letters with images of the dataset and with our own.

# CNN Model
# Confusion Matrix



# CNN Model Predictions With Data Set Images




# Our Pictures

Label = T

Label = A

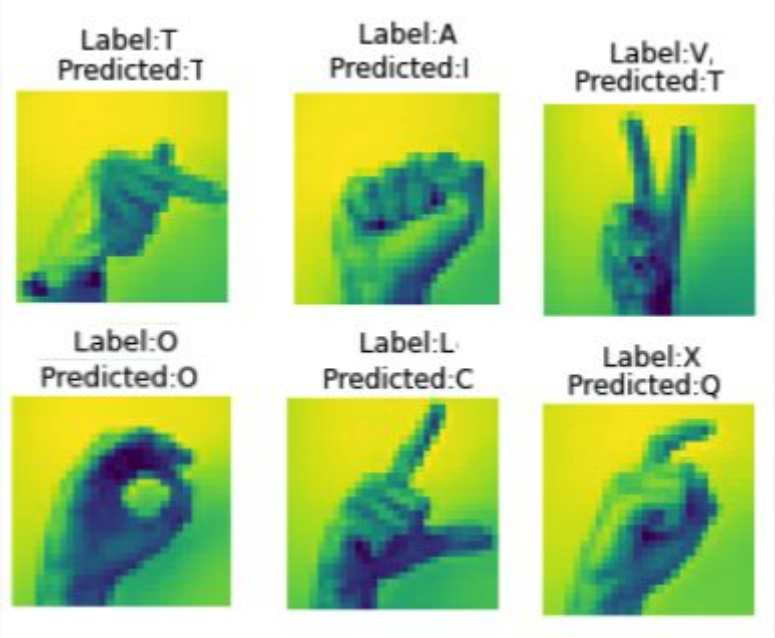
Label = V

Label = O

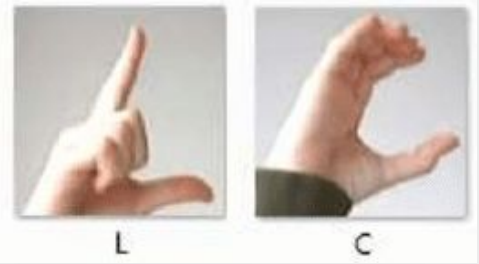
Label = L

Label = X

# Logistic Regression Model Predictions With Our Pictures



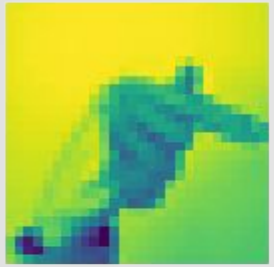




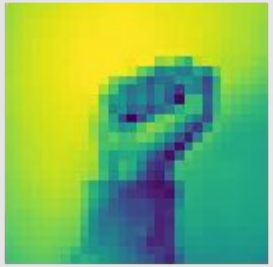


# CNN Model Predictions With Our Pictures

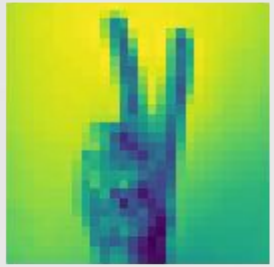
Label = T
Predicted = T



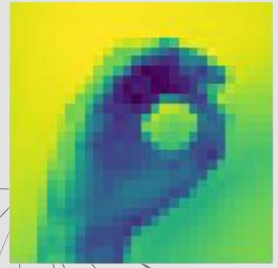
Label = A
Predicted = N



Label = V
Predicted = V

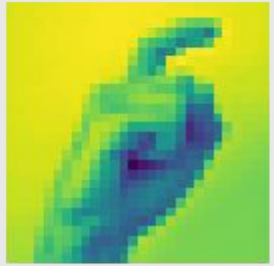


Label = O
Predicted = O



Label = L
Predicted = L

Label = X
Predicted = X



# 06

# Conclusion

How can we improve?