Project 1

Sign-Language MNIST and Classification

FAA 2024/25

Diogo Marto	108546
Pedro Azevedo	102567

Content

01

Dataset Description

02

State of the Art

03

Models

04

Comparison

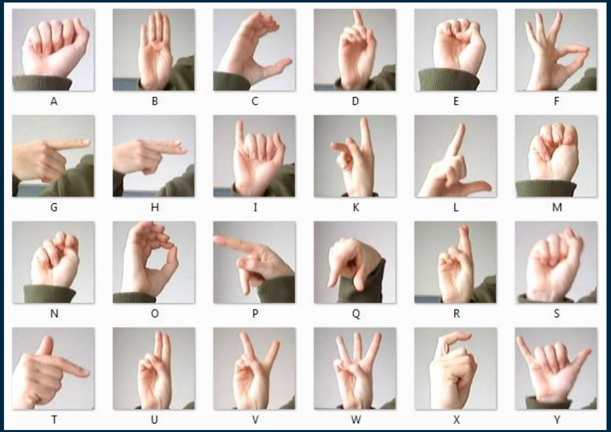
01 Dataset Description

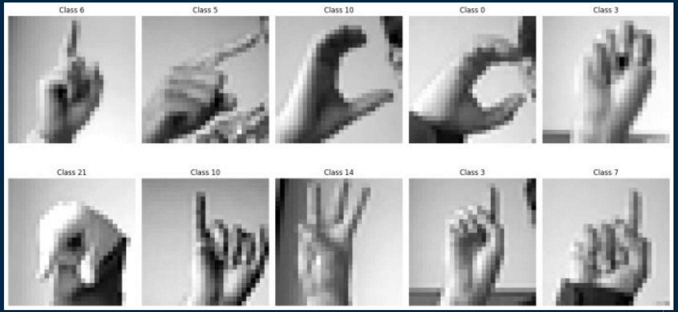
- Dataset is a transformation of another dataset

- Data was presplit into 2 files:

  Train: 27455 examples
  Test: 7172 examples

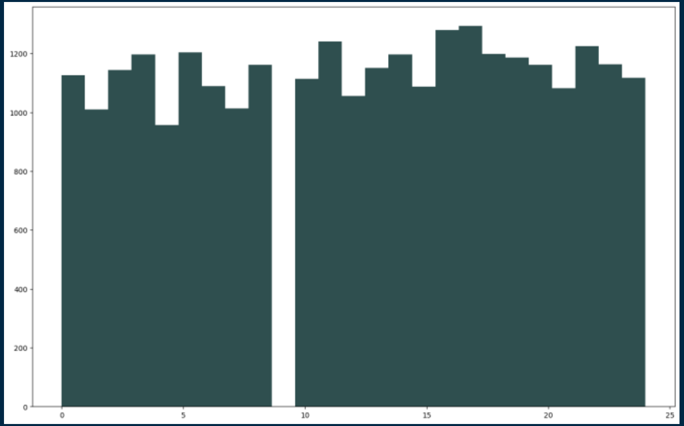
- 24 classes more or less evenly distributed



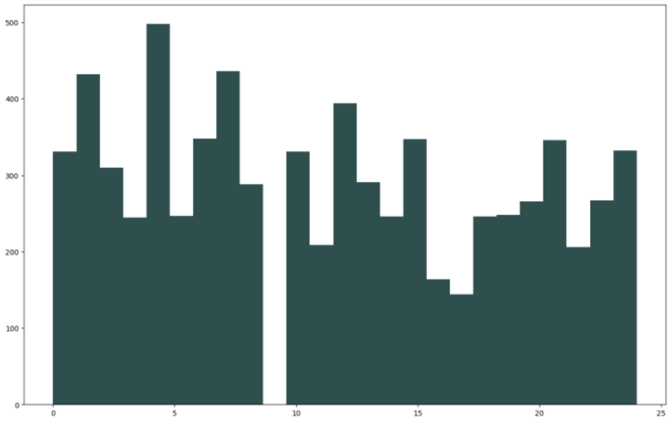


01 Dataset Description

Train



Test

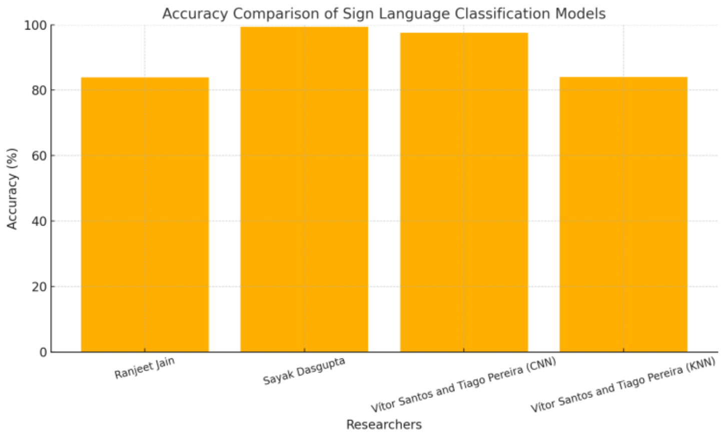


02 State of the art

- Ranjeet Jain using CNN has able to achieve 83.87% accuracy

- Sayak Dasgupta using CNN has able to achieve 99.40% accuracy but using data augmentation

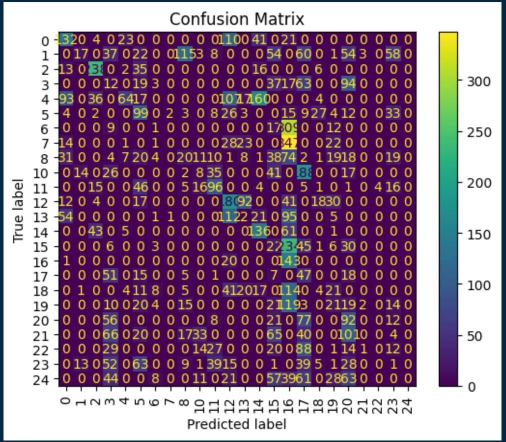
- Vitor Santos and Tiago Pereira were able achieve 84% accuracy using a KNN and 97% using a CNN

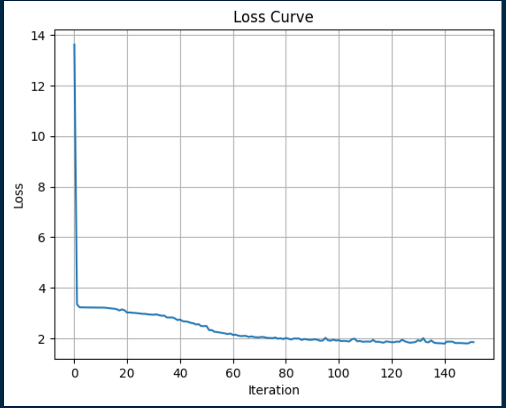


Models

03 Models – NN 1

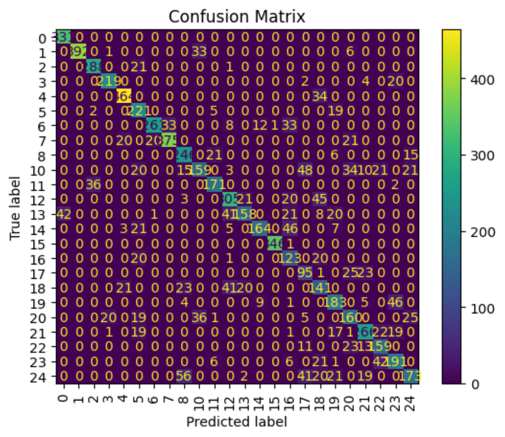
- Model built using most basic parameters, to understand how a simple model will perform
- No data normalization
- No L2 regularization term
- Accuracy 18,61% F1-Score 0.156



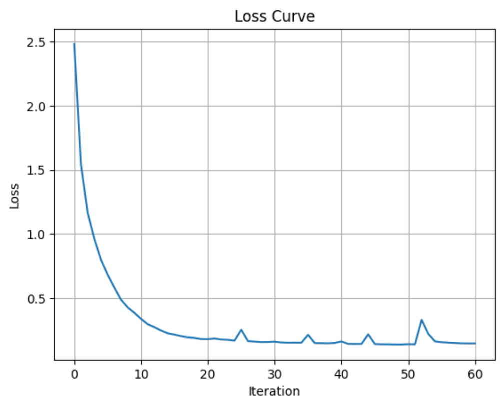


03 Models – NN 2

- Added data normalization
- Ran simulations to learn best hidden layer structure (1st 256 , 2nd 128)
- Shows clear improvements comparing to Model 1
- Accuracy 77.96% F1-Score 0.7577

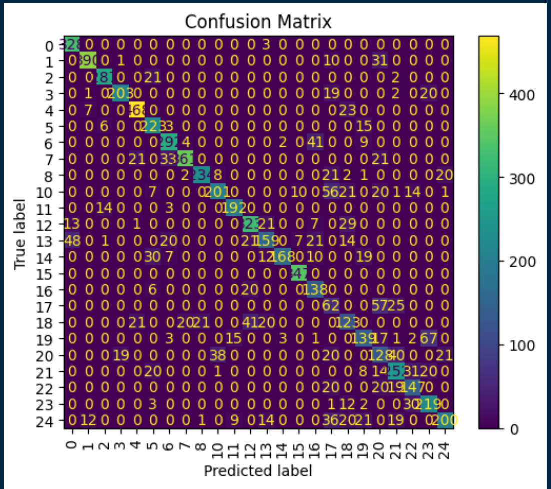


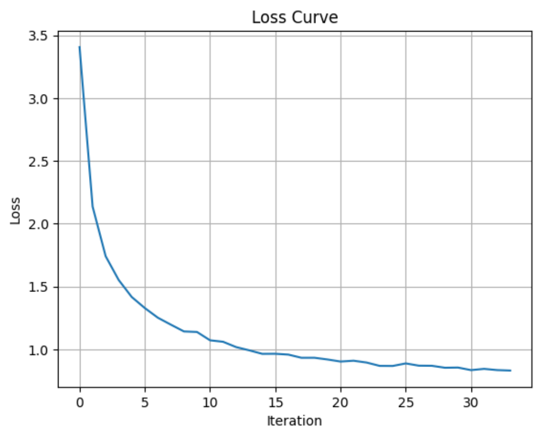
	0	1	2	3	4	5	6	7	8	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	331	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	392	0	1	0	0	0	0	0	33	0	0	0	0	0	0	0	0	0	6	0	0	0	0
2	0	0	288	0	0	21	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	219	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	4	0	20	0
4	0	0	0	0	464	0	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	0	0	0
5	0	0	2	0	0	221	0	0	0	0	5	0	0	0	0	0	0	0	19	0	0	0	0	0
6	0	0	0	0	0	0	261	33	0	0	0	8	0	12	1	33	0	0	0	0	0	0	0	0
7	0	0	0	0	20	0	20	375	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0
8	0	0	0	0	0	0	0	0	246	0	21	0	0	0	0	0	0	0	6	0	0	0	0	15
10	0	0	0	0	0	20	0	0	15	159	0	3	0	0	0	0	48	0	0	34	10	21	0	21
11	0	0	36	0	0	0	0	0	0	0	171	0	0	0	0	0	0	0	0	0	0	0	2	0
12	0	0	0	0	0	0	0	0	3	0	0	305	21	0	0	20	0	45	0	0	0	0	0	0
13	42	0	0	0	0	0	1	0	0	0	0	41	158	0	0	21	0	8	20	0	0	0	0	0
14	0	0	0	0	3	21	0	0	0	0	0	5	0	164	0	46	0	0	7	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	346	1	0	0	0	0	0	0	0	0
16	0	0	0	0	0	20	0	0	0	0	0	1	0	0	0	123	0	20	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	95	1	0	25	23	0	0	0
18	0	0	0	0	21	0	0	0	23	0	0	41	20	0	0	0	0	141	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	4	0	0	0	0	9	0	1	0	0	183	0	5	0	46	0
20	0	0	0	20	0	19	0	0	0	36	1	0	0	0	0	0	5	0	0	160	0	0	0	25
21	0	0	0	1	0	19	0	0	0	0	0	0	0	0	0	1	0	0	17	1	266	22	19	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	23	13	159	0	0
23	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	6	0	21	1	0	0	42	191	0
24	0	0	0	0	0	0	0	0	56	0	0	0	2	0	0	0	41	20	21	0	19	0	0	173



03 Models – NN 3

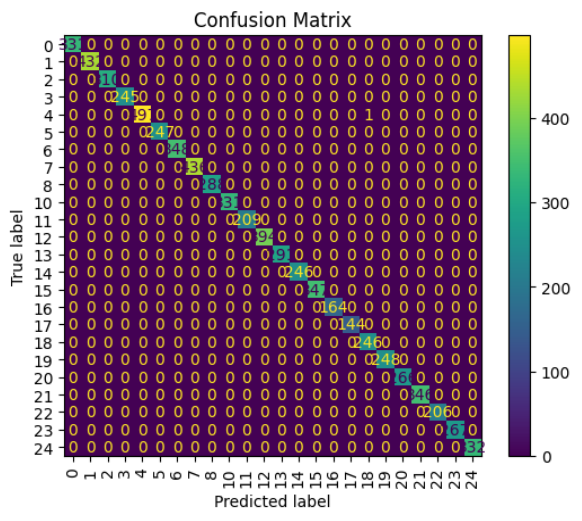
- Added L2 regularization term (alpha)
- Ran simulations to get the best alpha value (0.1)
- Accuracy 77.86% F1-Score 0.7559
- Overfitting still present

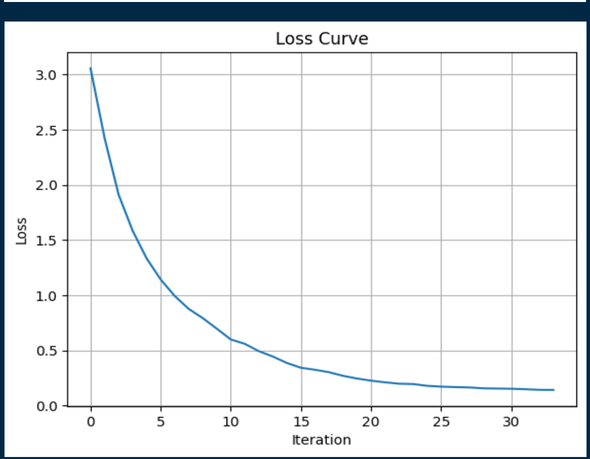




03 Models – NN 4

- Added data augmentation in both training and testing
- Model performance shows clear improvement
- Accuracy 99.48%
- F1-Score 0.9979

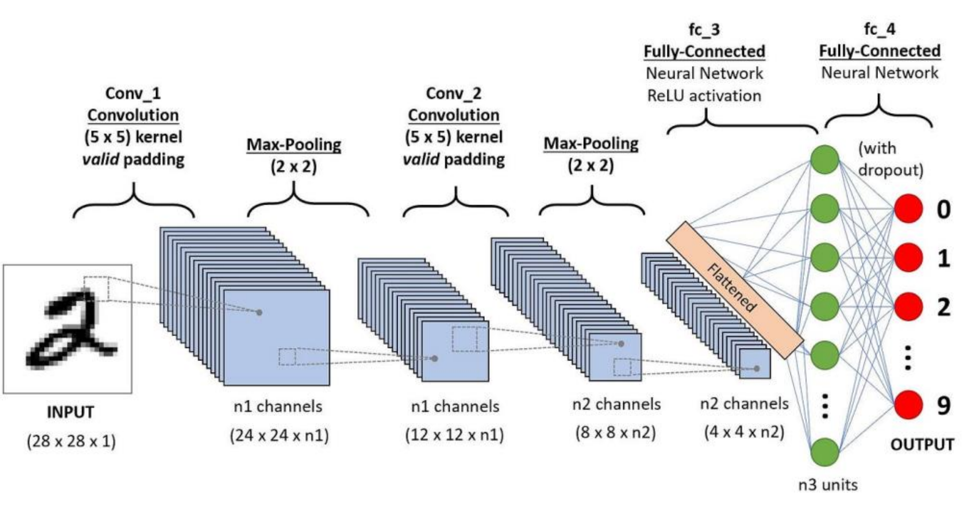




03 Models – CNN What is ?

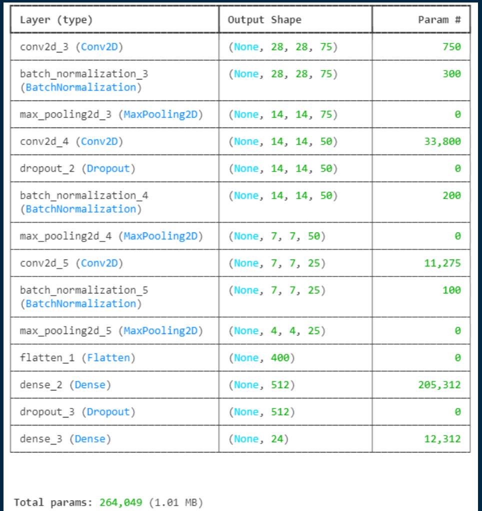
- Very Good at dealing with images as it makes use of convolutions.

- From the references we know that they perform well on this problem



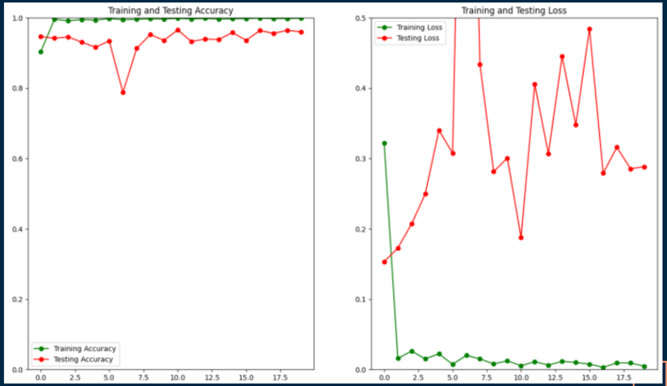
03 Models – CNN 1

- Based on references we built the architecture
- Trained without Data Augmentation and Adaptive learning rate
- Accuracy 90.90%
- F1-score 0.9019



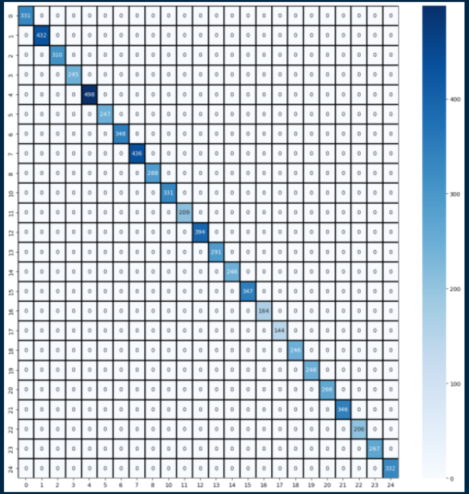
Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 28, 28, 75)	750
batch_normalization_3 (BatchNormalization)	(None, 28, 28, 75)	300
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 75)	0
conv2d_4 (Conv2D)	(None, 14, 14, 50)	33,800
dropout_2 (Dropout)	(None, 14, 14, 50)	0
batch_normalization_4 (BatchNormalization)	(None, 14, 14, 50)	200
max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 50)	0
conv2d_5 (Conv2D)	(None, 7, 7, 25)	11,275
batch_normalization_5 (BatchNormalization)	(None, 7, 7, 25)	100
max_pooling2d_5 (MaxPooling2D)	(None, 4, 4, 25)	0
flatten_1 (Flatten)	(None, 400)	0
dense_2 (Dense)	(None, 512)	205,312
dropout_3 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 24)	12,312

Total params: 264,049 (1.01 MB)

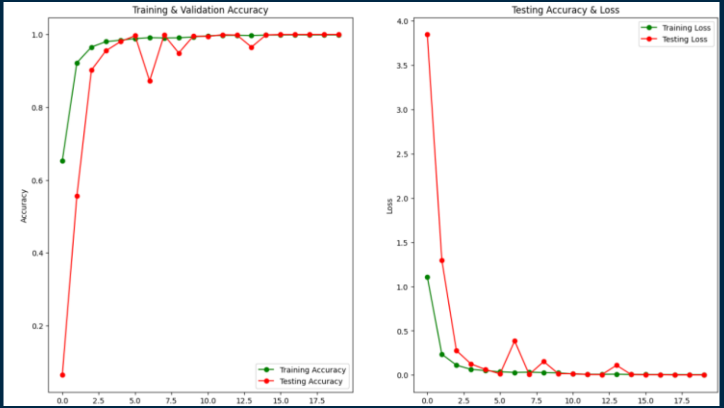


03 Models – CNN 1

- Same Architecture
- But using data augmentation and adaptive learning rate
- Accuracy 100%
- F1-score 1

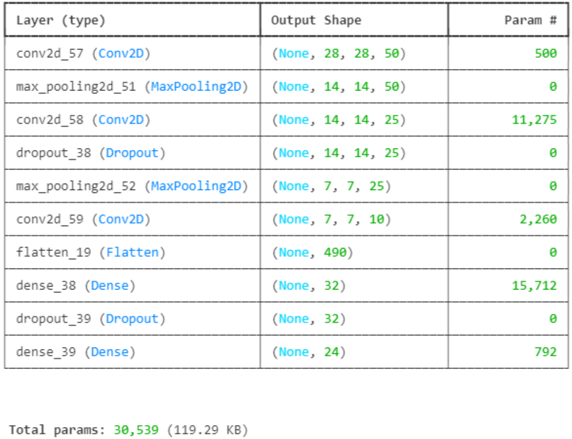


	0	1	2	3	4	5	6	7	8	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	331	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	432	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	310	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	245	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	498	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	247	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	348	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	436	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	288	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	331	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	209	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	394	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	291	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	246	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	347	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	164	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	144	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	246	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	248	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	266	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	346	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	206	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	267	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	332



03 Models – CNN 2

- Smaller model to not overfit without data augmentation
- L2 regularization on dense layers and strength 0.01
- Architecture based on iterations where we adjusted model size and alpha strength

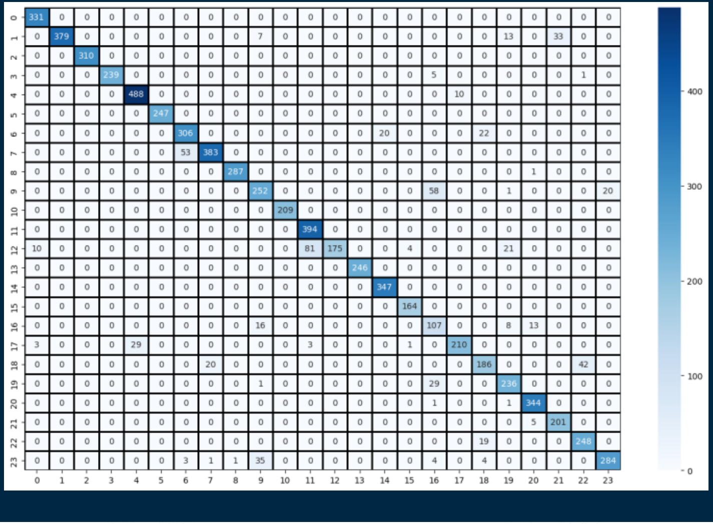


Layer (type)	Output Shape	Param #
conv2d_57 (Conv2D)	(None, 28, 28, 50)	500
max_pooling2d_51 (MaxPooling2D)	(None, 14, 14, 50)	0
conv2d_58 (Conv2D)	(None, 14, 14, 25)	11,275
dropout_38 (Dropout)	(None, 14, 14, 25)	0
max_pooling2d_52 (MaxPooling2D)	(None, 7, 7, 25)	0
conv2d_59 (Conv2D)	(None, 7, 7, 10)	2,260
flatten_19 (Flatten)	(None, 490)	0
dense_38 (Dense)	(None, 32)	15,712
dropout_39 (Dropout)	(None, 32)	0
dense_39 (Dense)	(None, 24)	792

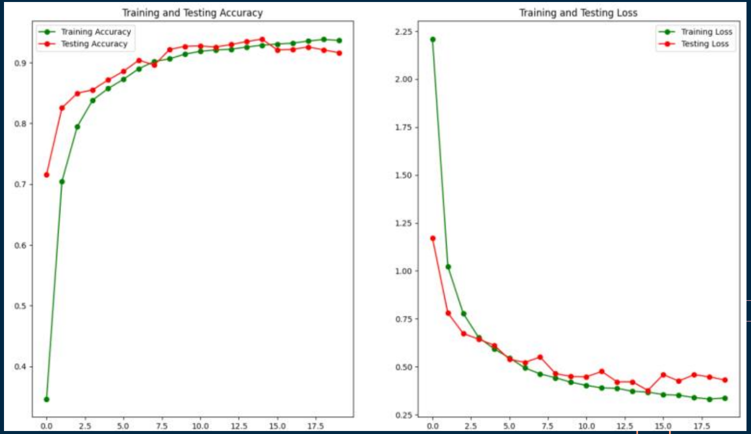
Total params: 30,539 (119.29 KB)

03 Models – CNN 2

- Loss curve shows that the model didn't overfit
- Accuracy is less than model 1 with data augmentation
- Hardest classes are 11 and 16
- Accuracy 91.64%
- F1-score 0.9106

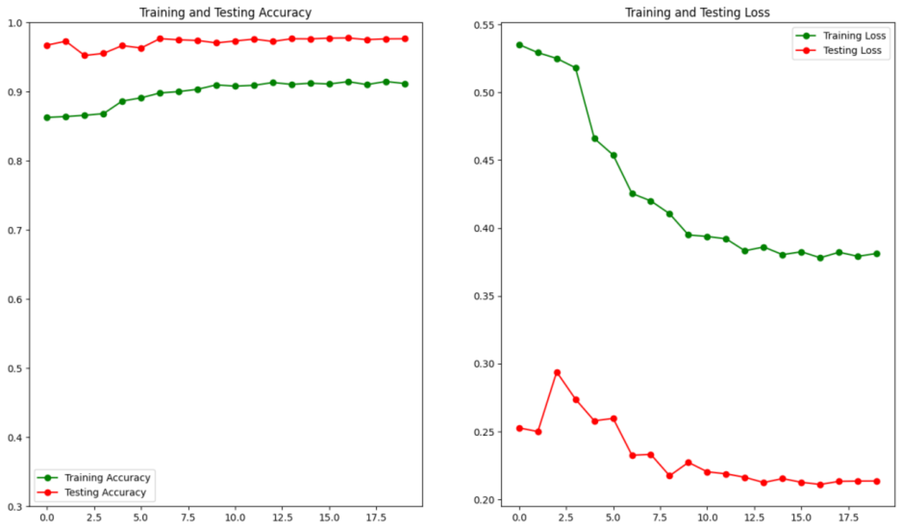


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	331	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	379	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	13	0	33	0	0
2	0	0	310	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	239	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	1	0
4	0	0	0	0	488	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
5	0	0	0	0	0	247	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	306	0	0	0	0	0	0	0	20	0	0	0	22	0	0	0	0	0
7	0	0	0	0	0	0	53	383	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	287	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
9	0	0	0	0	0	0	0	0	0	252	0	0	0	0	0	0	58	0	0	1	0	0	0	20
10	0	0	0	0	0	0	0	0	0	0	209	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	394	0	0	0	0	0	0	0	0	0	0	0	0
12	10	0	0	0	0	0	0	0	0	0	0	81	175	0	0	4	0	0	0	21	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	246	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	347	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	164	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	107	0	0	8	13	0	0	0
17	3	0	0	0	29	0	0	0	0	0	0	3	0	0	0	1	0	210	0	0	0	0	0	0
18	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	186	0	0	0	42	0
19	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	29	0	0	236	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	344	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	201	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	248	0
23	0	0	0	0	0	0	3	1	1	35	0	0	0	0	0	0	4	0	4	0	0	0	0	284



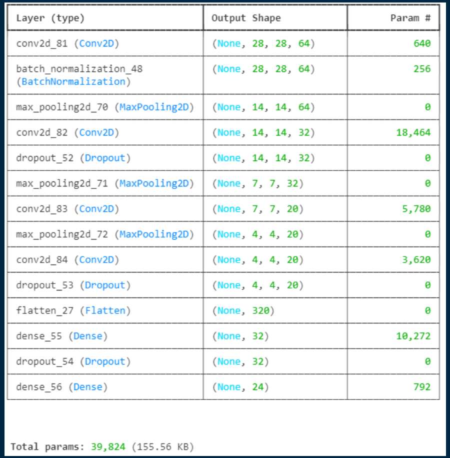
03 Models – CNN 2

- Using data augmentation on this model gives an interesting result where train loss > test loss
- Data augmentation introduced patterns than don’t occur on test.
- Accuracy 97.65%
- F1-score 0.9737



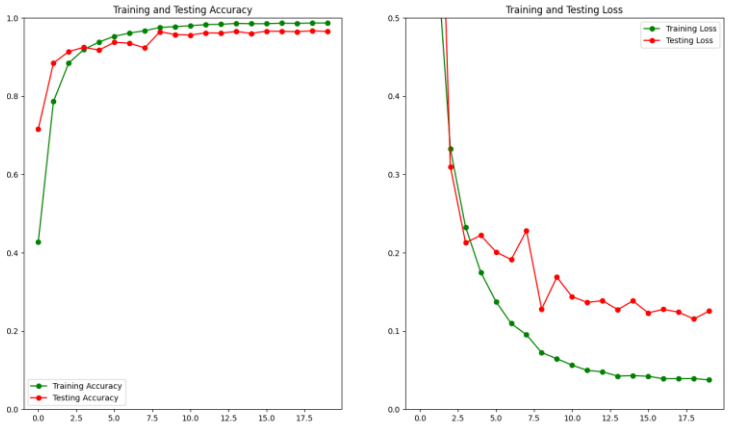
03 Models – CNN3

- Final model meant to be small and precise and with data augmentation in mind
- Needed 3 layers of convolution
- Accuracy 96.52%
- F1-score 0.9660



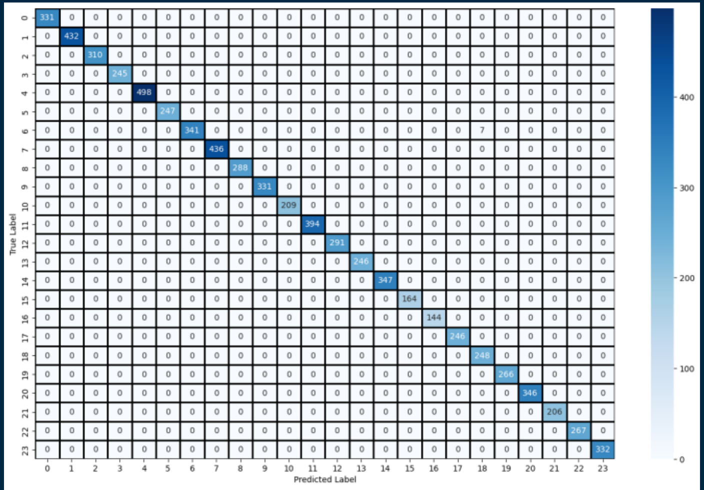
Layer (type)	Output Shape	Param #
conv2d_81 (Conv2D)	(None, 28, 28, 64)	640
batch_normalization_48 (BatchNormalization)	(None, 28, 28, 64)	256
max_pooling2d_70 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_82 (Conv2D)	(None, 14, 14, 32)	18,464
dropout_52 (Dropout)	(None, 14, 14, 32)	0
max_pooling2d_71 (MaxPooling2D)	(None, 7, 7, 32)	0
conv2d_83 (Conv2D)	(None, 7, 7, 20)	5,780
max_pooling2d_72 (MaxPooling2D)	(None, 4, 4, 20)	0
conv2d_84 (Conv2D)	(None, 4, 4, 20)	3,620
dropout_53 (Dropout)	(None, 4, 4, 20)	0
flatten_27 (Flatten)	(None, 320)	0
dense_55 (Dense)	(None, 32)	10,272
dropout_54 (Dropout)	(None, 32)	0
dense_56 (Dense)	(None, 24)	792

Total params: 39,824 (155.56 KB)

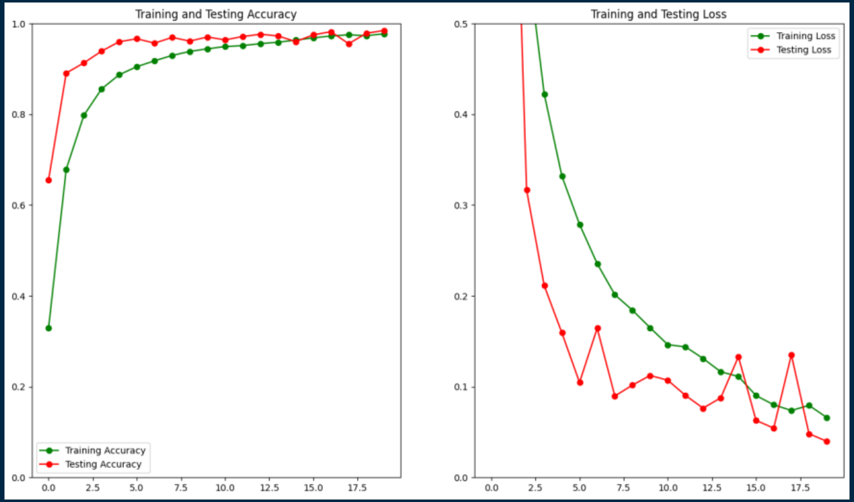


03 Models – CNN3

Accuracy:99.90% F1-score:0.9989

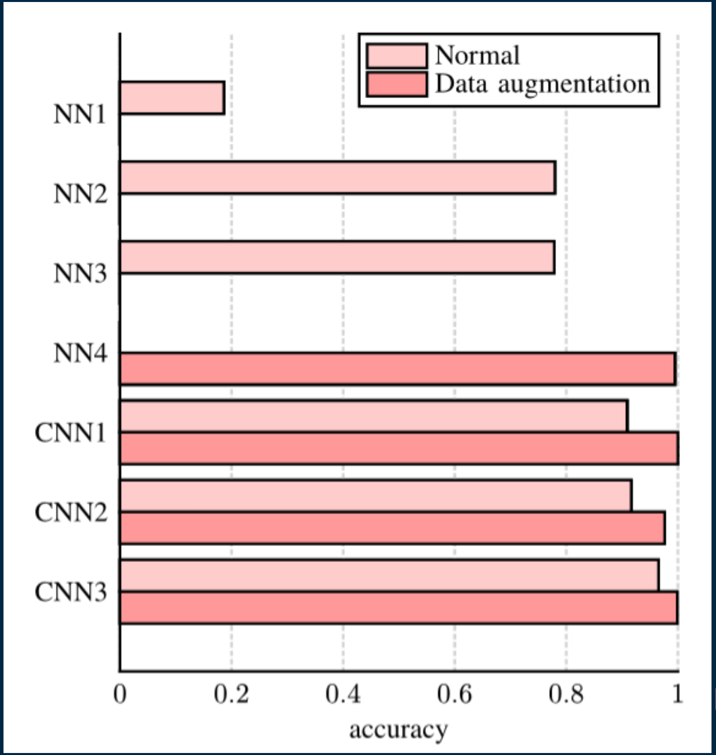


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	331	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	432	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	310	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	245	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	498	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	247	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	341	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0
7	0	0	0	0	0	0	0	436	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	288	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	331	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	209	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	394	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	291	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	246	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	347	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	164	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	144	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	246	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	248	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	266	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	346	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	206	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	267	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	332



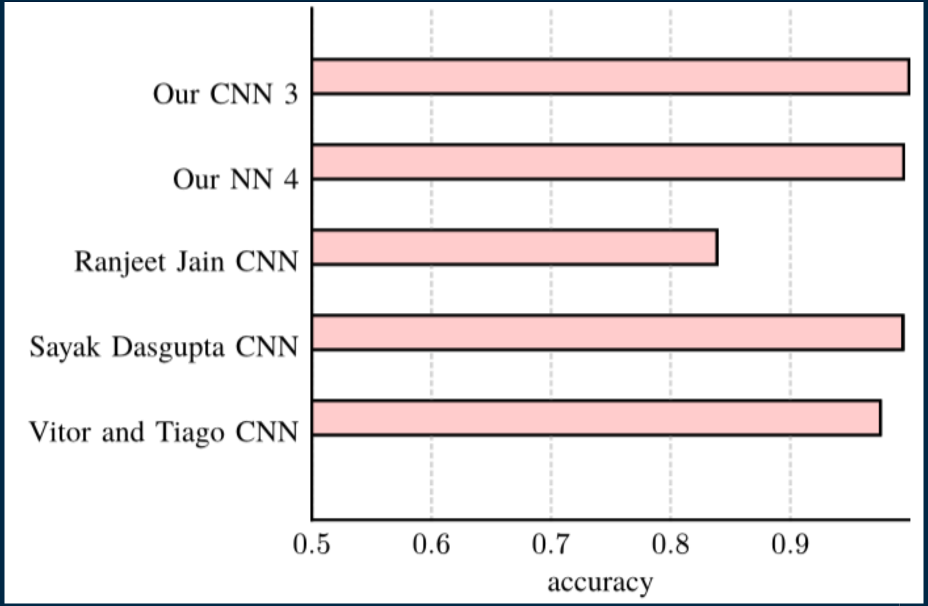
04 Comparison

- Without data augmentation the CNNs performed much better
- Data augmentation had a big effect on this problem



04 Comparison

- Our CNN 3 has 8x less parameters than the best model
- Our NN 4 performed better than we expected



Conclusion

- Data augmentation had a large impact both in CNN and ANN.
- Both approaches produced very good models.
- For further improvements we could experiment with PCA and ensemble of small models.

References:

- [1] R. Jain, “Deep Learning Using Sign Language.” [Online]. Available: https://www.kaggle.com/code/ranjeetjain3/deep-learning-usingsign-langugage
- [2] S. Dasgupta, “Sign Language Classification - CNN (99.40% Accuracy).” [Online]. Available: https://www.kaggle.com/code/sayakdasgupta/ sign-language-classification-cnn-99-40-accuracy
- [3] V. Santos and T. Pereira, “Exploring Models for Sign Language Classification.” Aveiro, Portugal, 2023.

THANK YOU