Using ML to Read the ASL Alphabet

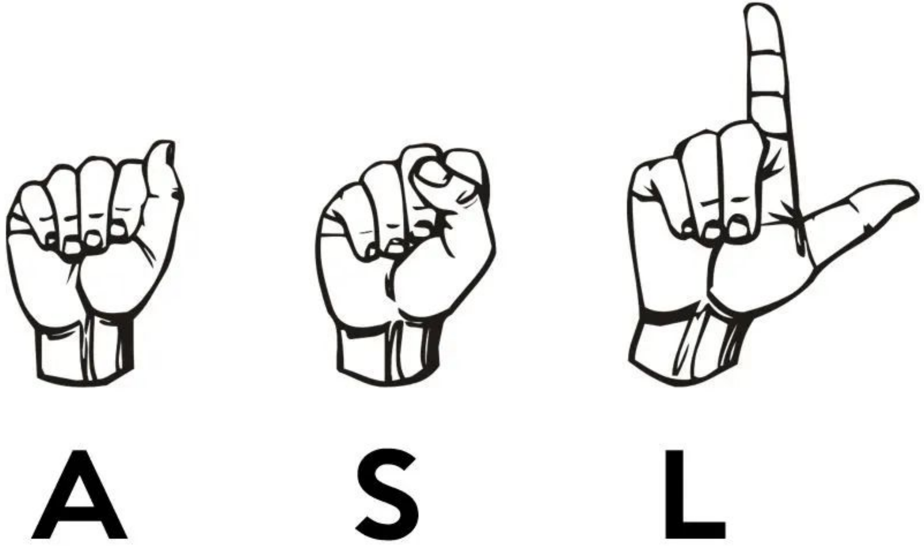
A lesson in preprocessing data

Emilie Nielsen, Morten Bendtsen,
Ludvig Marcussen & Michelle Rix

UNIVERSITY OF COPENHAGEN

Introduction

- Classification of images of ASL alphabet
- Inspired by kaggle competition
  - Data obtained from kaggle



Introduction to Data

1. The online data
- 29 categories
- 1 hand
- 3000 jpgs in each
- 240 X 240 pixels

A

B

C

D

....

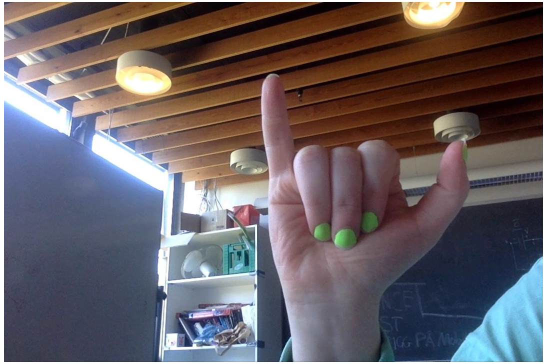
Delete

Space

Nothing

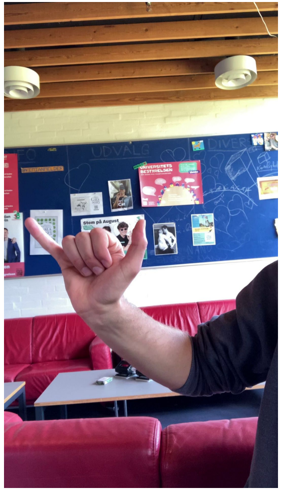
Introduction to Data

1. The online data
   - 29 categories
   - 1 hand
   - 3000 jpgs of each sign
   - 240 X 240 pixels



2. Self produced data
   - 29 categories
   - 4 hands
   - 960 jpgs of each sign
   - variable dimensions





Overview

ASL Images

Original images

- UMAP
- CNN
- Supervised clustering

Cropped images

- UMAP
- CNN
- Supervised clustering

Skeletonized images

- UMAP
- GNN
- Surprise

ASL Images

Original images	Cropped images	Skeletonized images
UMAP	UMAP	UMAP
CNN	CNN	GNN
Supervised clustering	Supervised clustering	Surprise

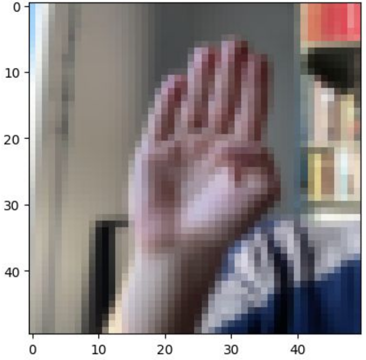
Preprocessing

A two step process

1. Compress photo to suitable format
2. Turn into array

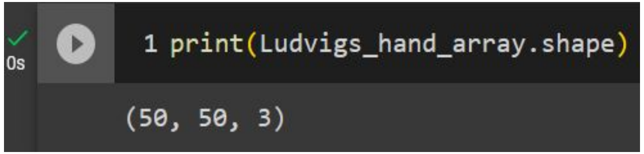
Original image

Compress



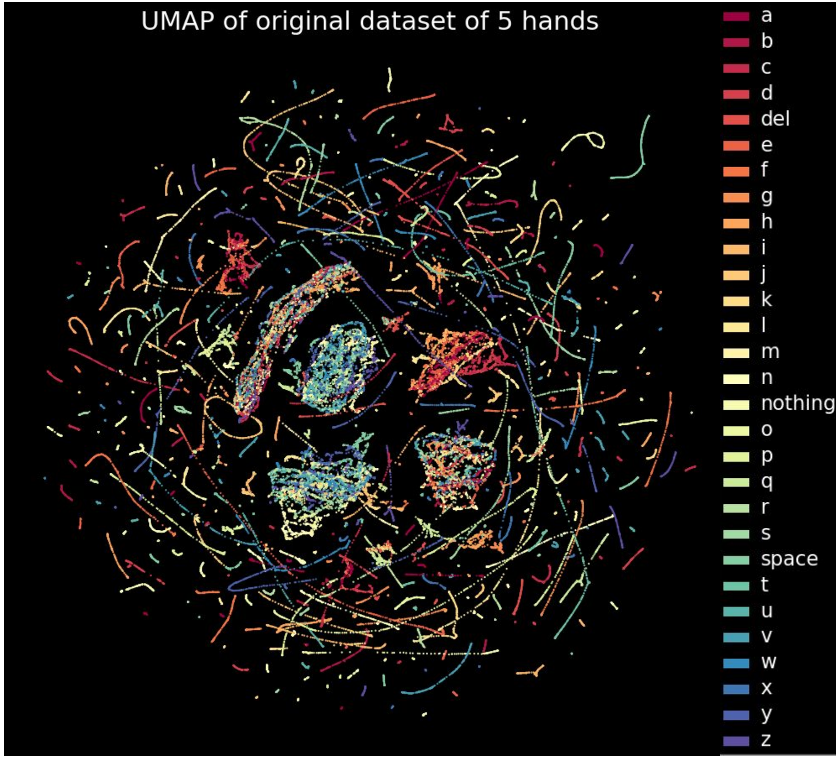
50X50 image

“Arraify”



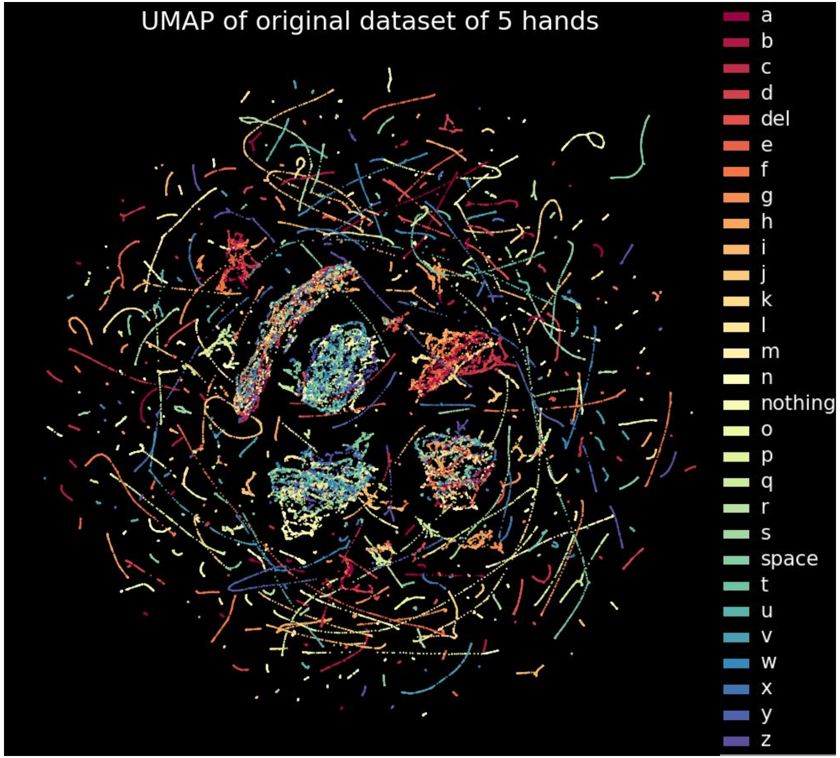
Dimensionality reduction using UMAP on original images

Colouring by sign

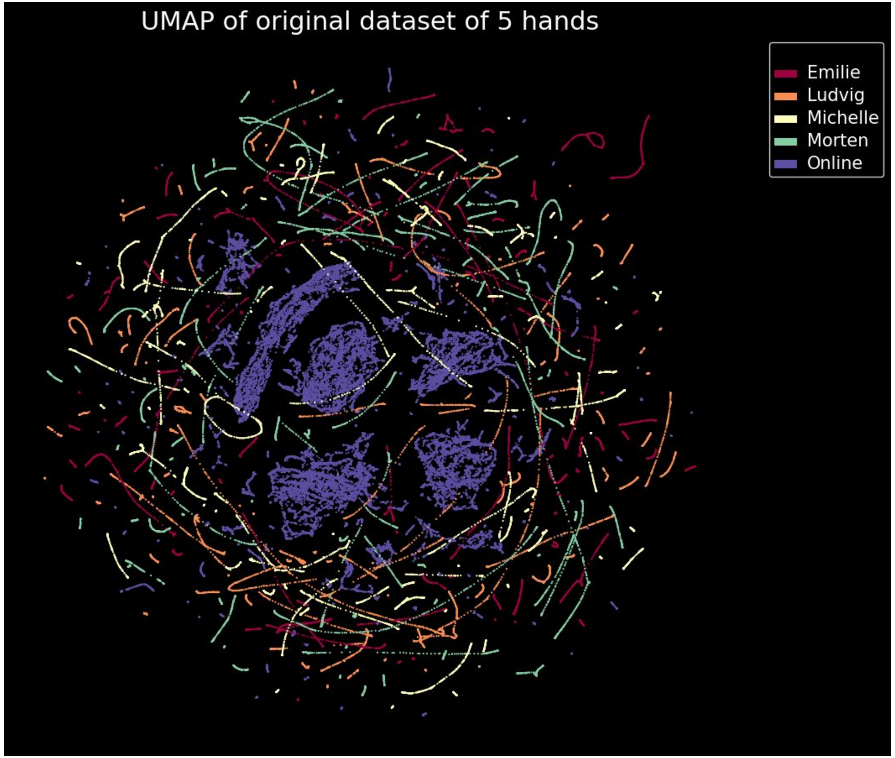


Dimensionality reduction using UMAP on original images

Colouring by sign

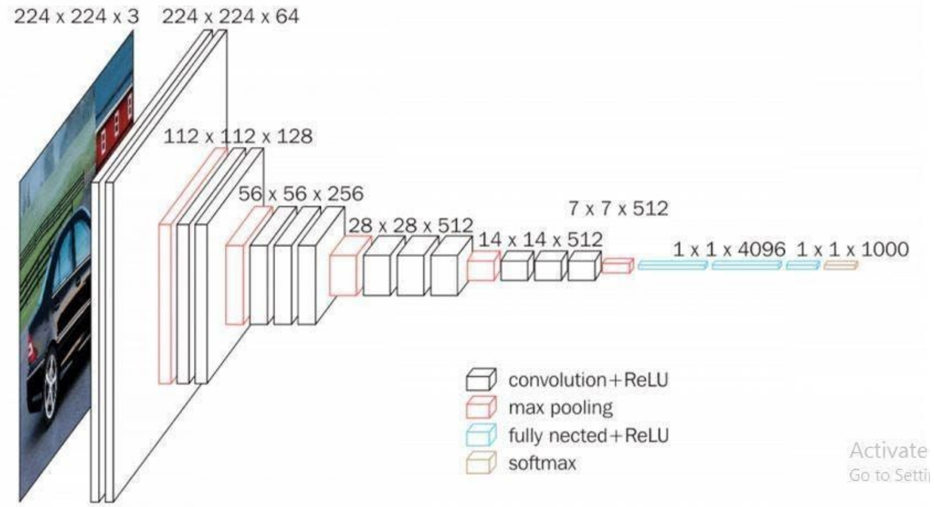


Colouring by people



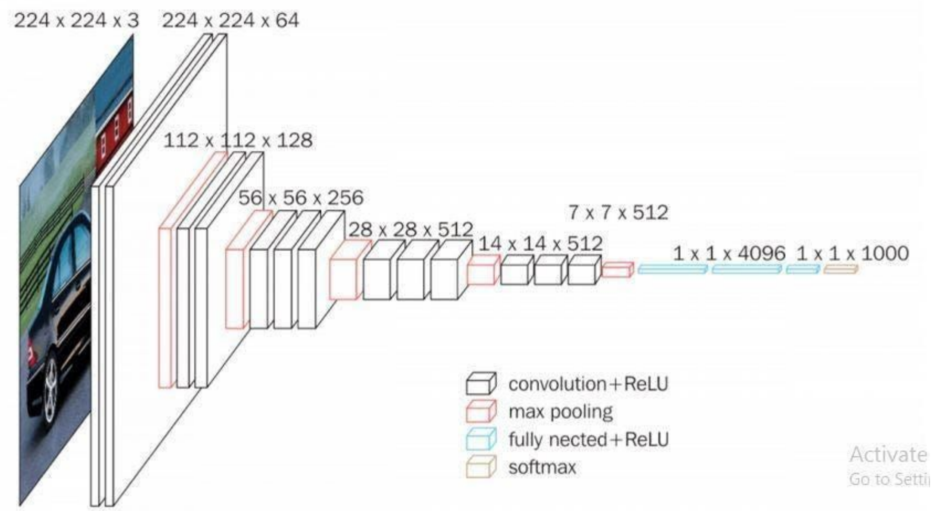
Using CNNs to classify signs

- VGG16 - a pretrained network

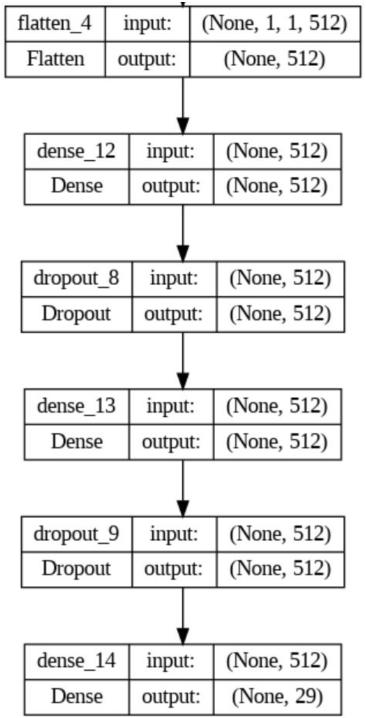


Using CNNs to classify signs

- VGG16 - a pretrained network

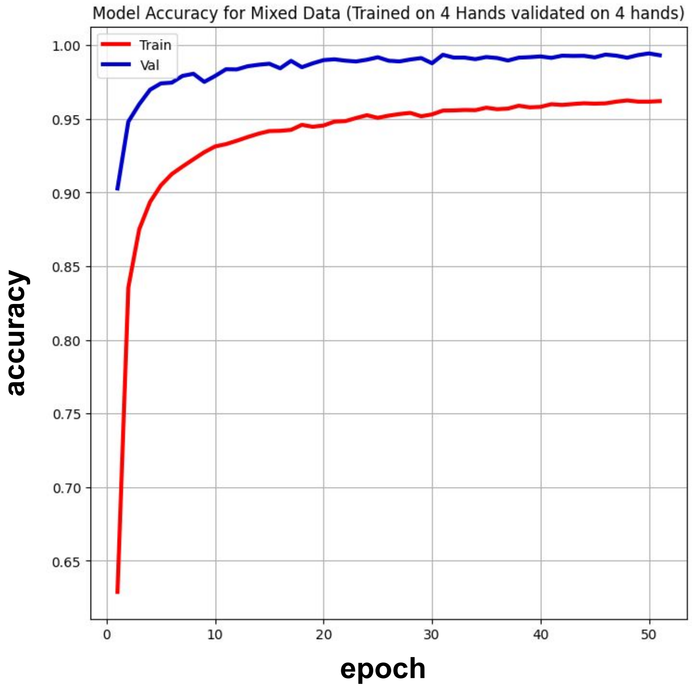


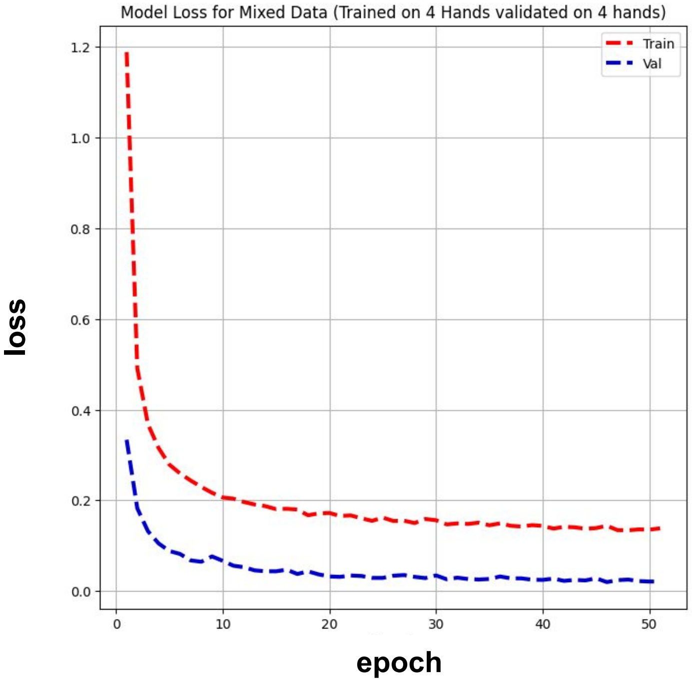
+



Using CNNs to classify signs

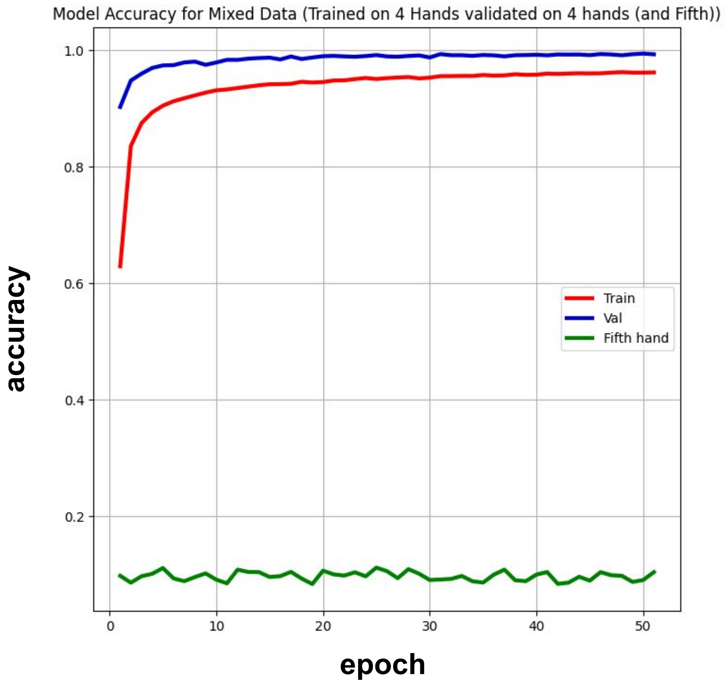
- Mixing the online data and 3 of our hands and doing train/test split

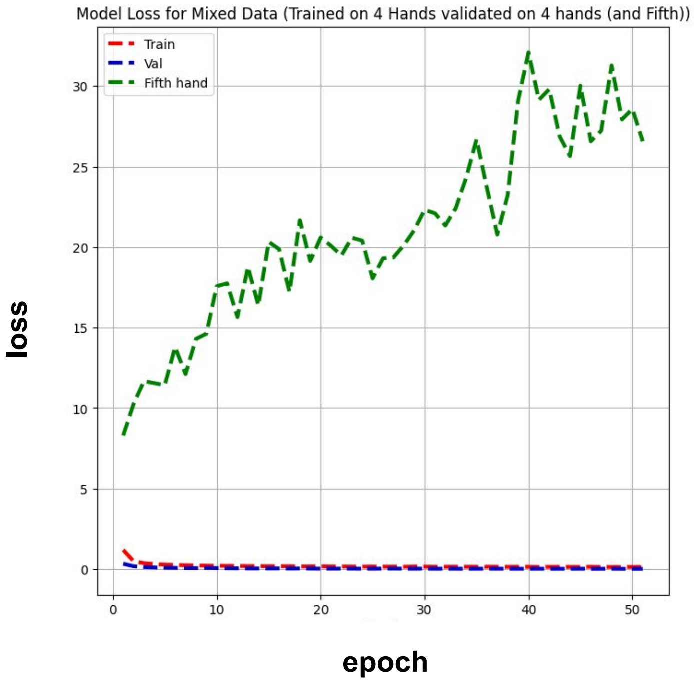




CNN classifies new signs with an accuracy of ~ 9 %

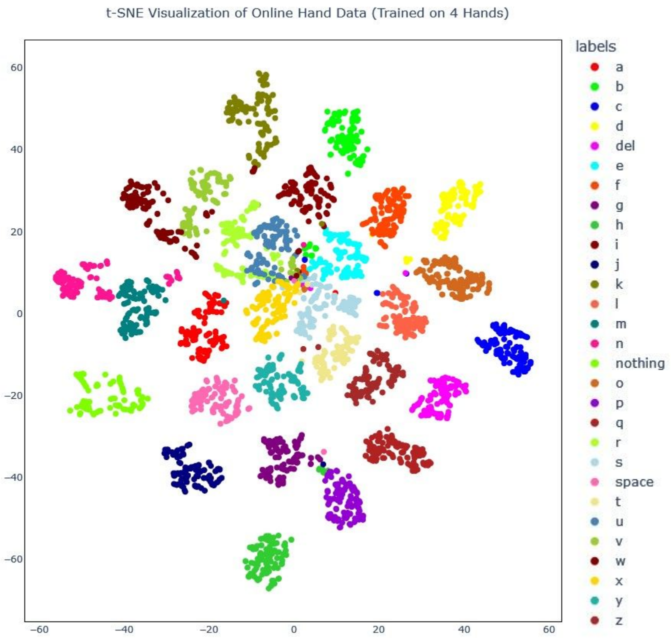
- Mixing the online data and 3 of our hands and doing train/test split
- What if we introduced a new hand?





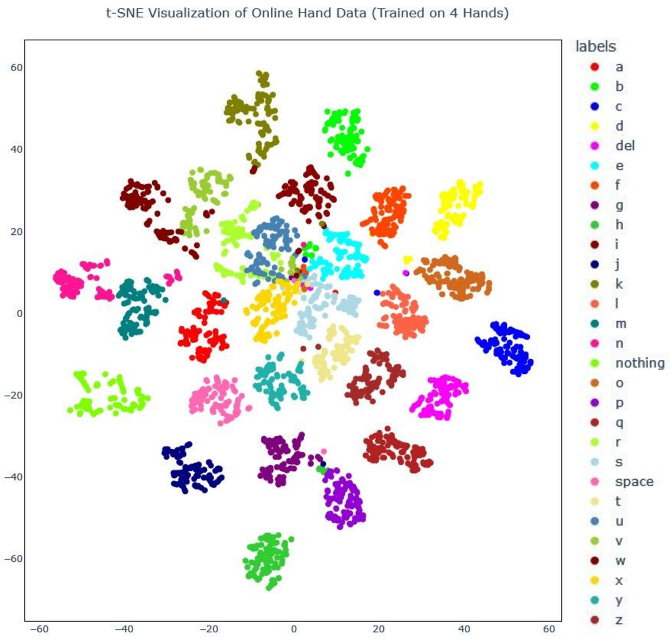
Visualizing Data Using Supervised Clustering

- Removing the output layer of CNN yields feature selection
- From these features we can perform t-sne to get a better look at the data!

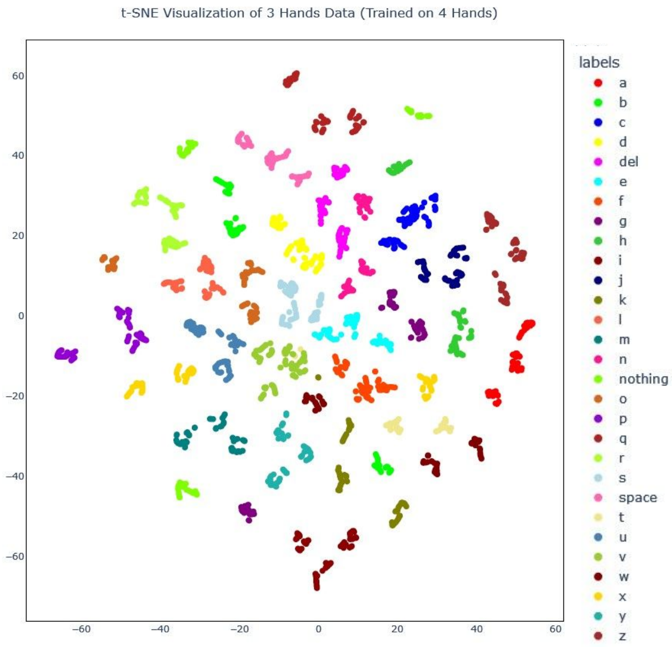


Visualizing Data Using Supervised Clustering

- Removing the output layer of CNN yields feature selection
- From these features we can perform t-sne to get a better look at the data!

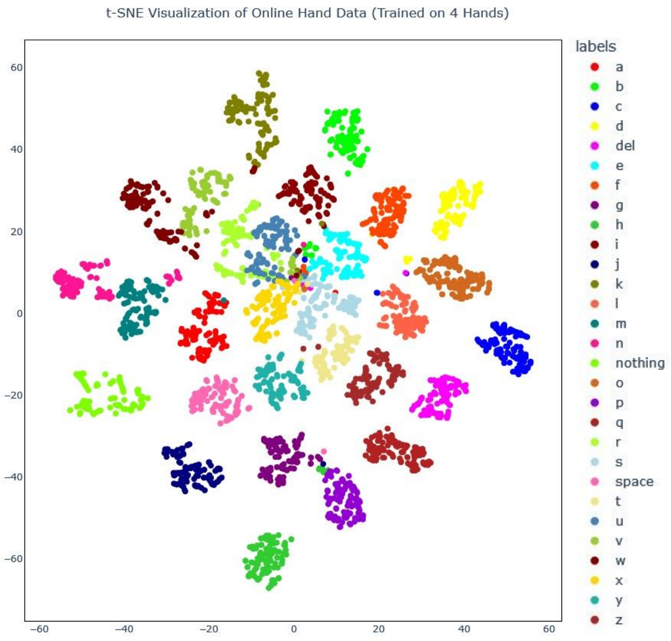


Our three hands →

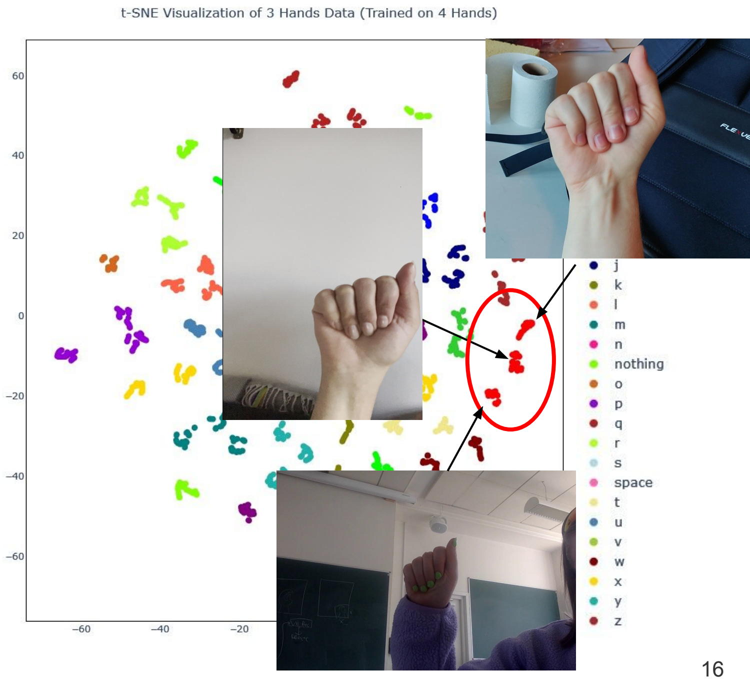


Visualizing Data Using Supervised Clustering

- Removing the output layer of CNN yields feature selection
- From these features we can perform t-sne to get a better look at the data!

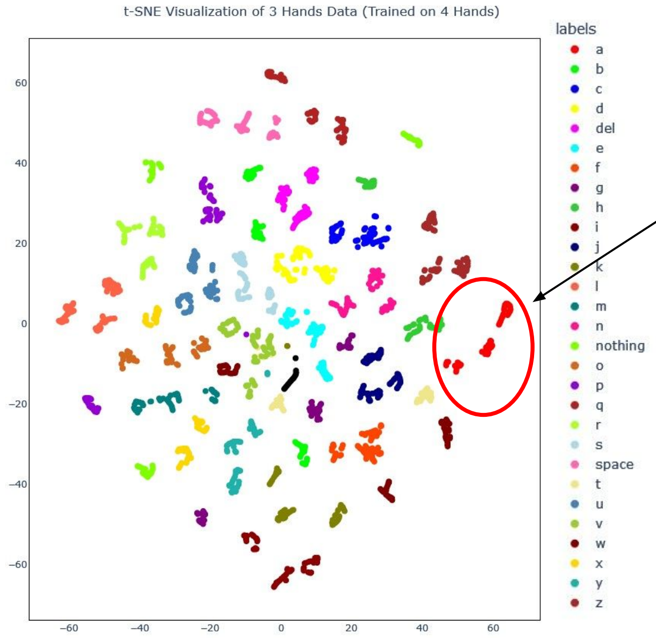


Our three hands



Visualizing Data Using Supervised Clustering

- What if we add a letter from a new hand?



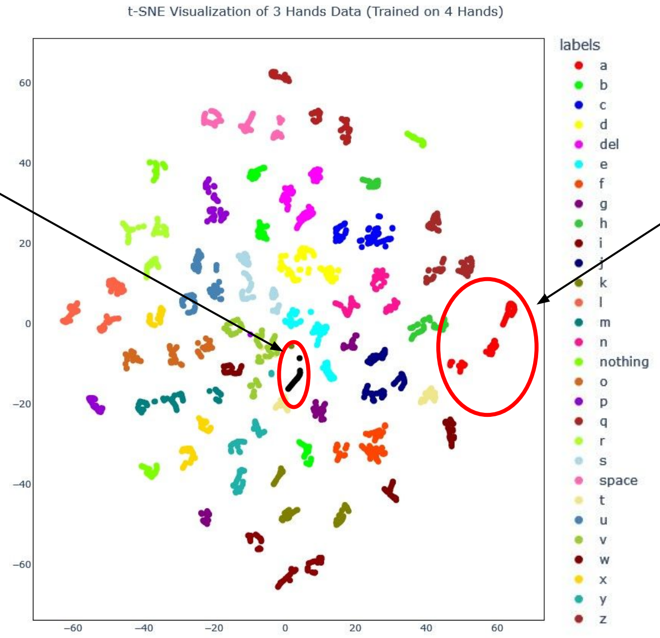
A’s that have been trained on

Visualizing Data Using Supervised Clustering

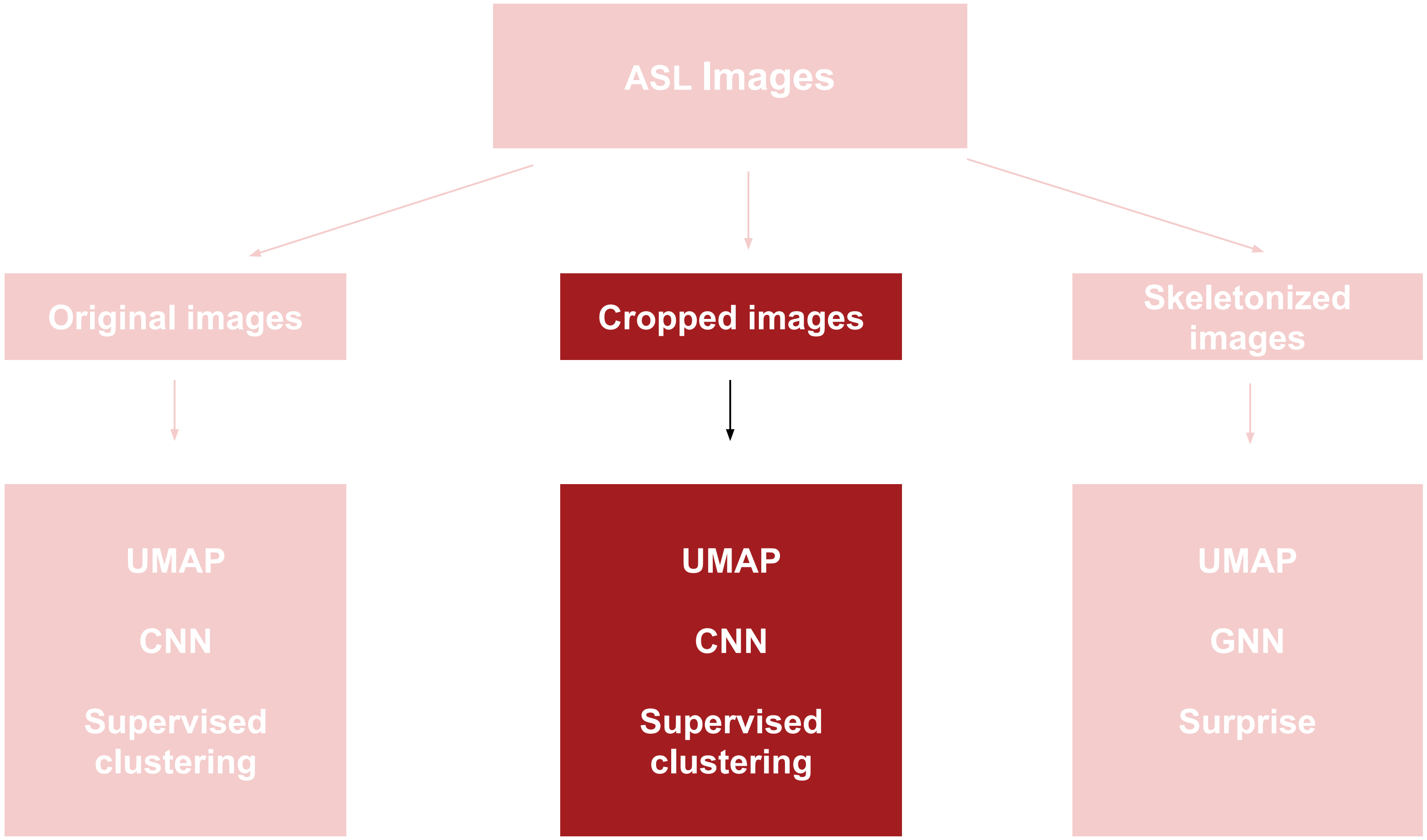
- What if we add a letter from a new hand?

A’s from new hand

A’s that have been trained on



Conclusion: The network sees background - not signs!


ASL Images
Original images
Cropped images
Skeletonized images
UMAP
CNN
Supervised clustering
UMAP
CNN
Supervised clustering
UMAP
GNN
Surprise

Introduction to cropped data

A three step process

1. Crop image to contain only the hand
2. Compress photo to suitable format
3. Turn into array

Original image

Crop

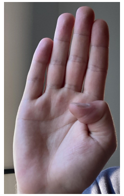
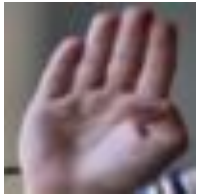


Image cutout

Compress



50X50 image

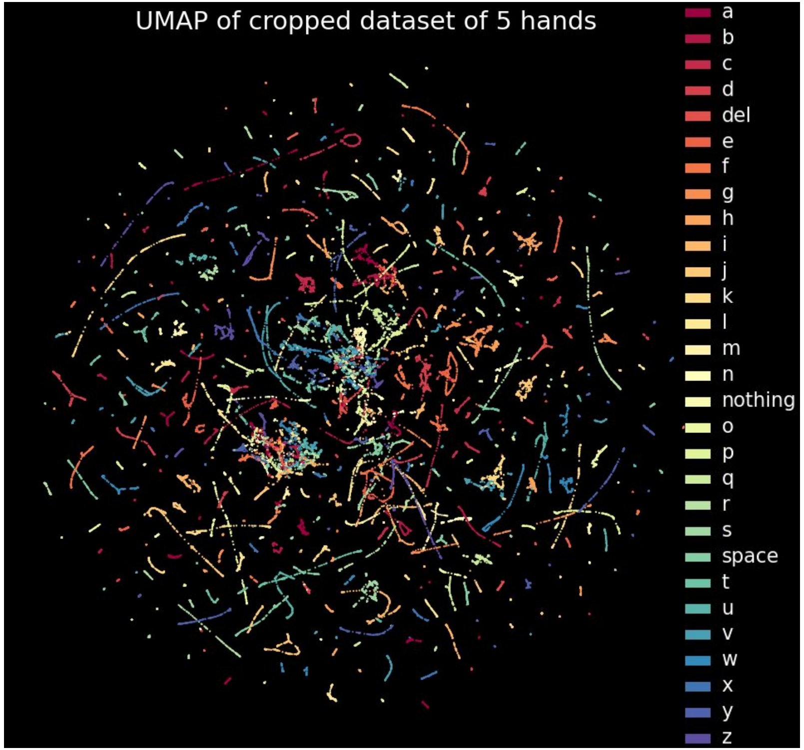
"Arraify"

```
1 print(Ludvigs_hand_array.shape)

(50, 50, 3)
```

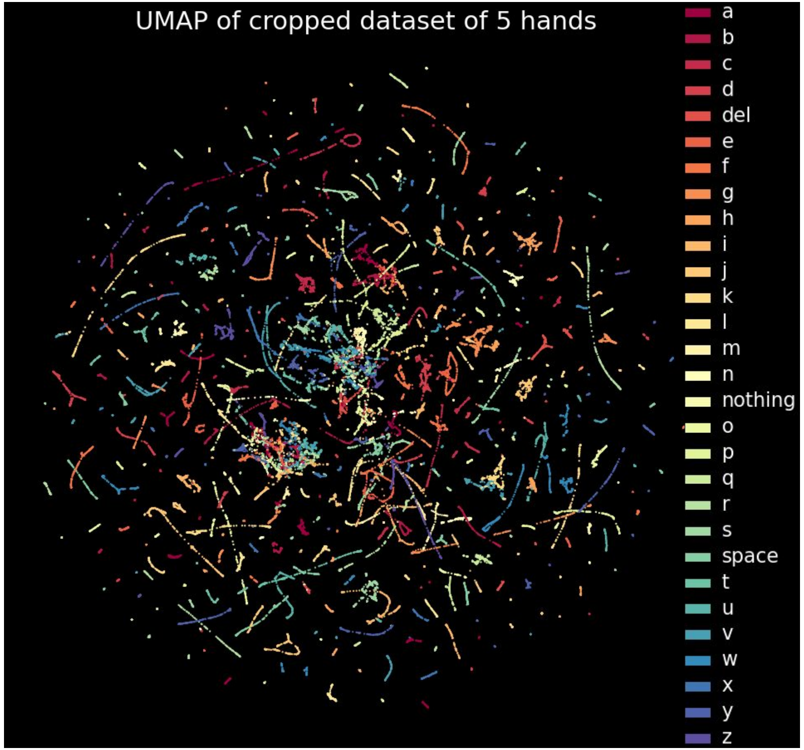
Dimensionality reduction using UMAP on cropped images

Colouring by sign (cropped)

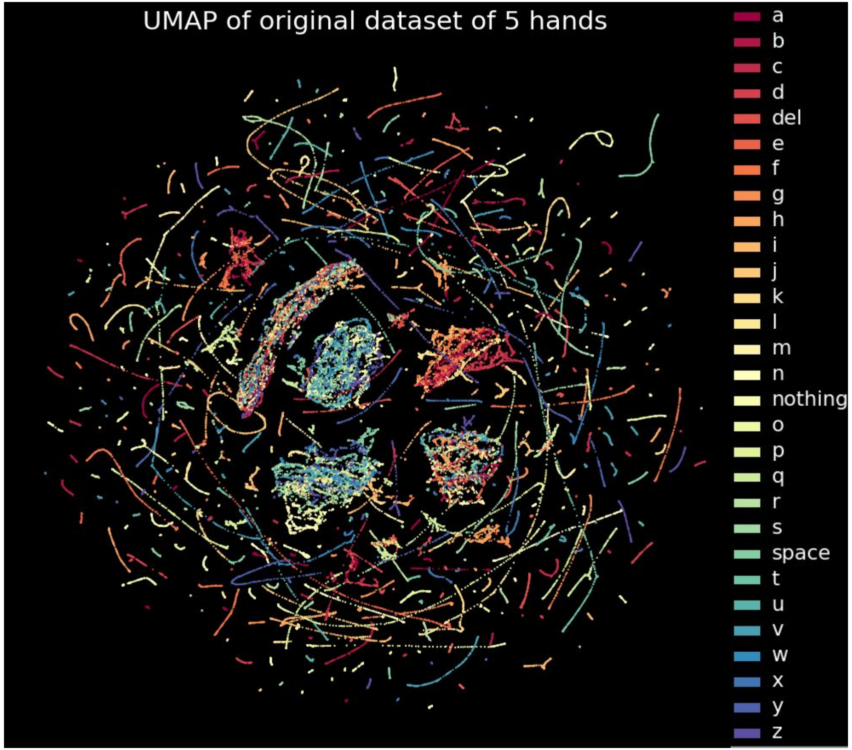


Dimensionality reduction using UMAP on cropped images

Colouring by sign (cropped)

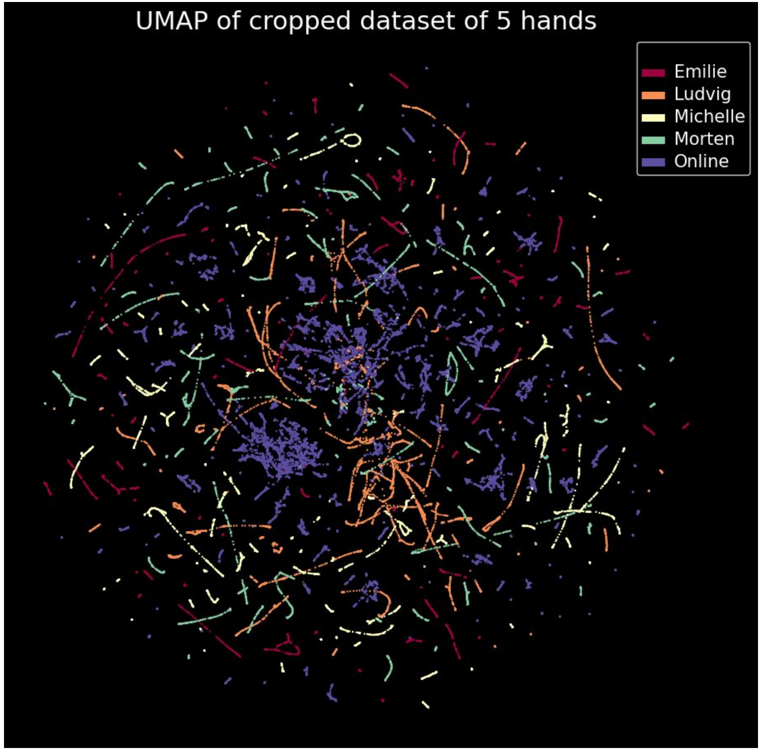


Colouring by sign (original)



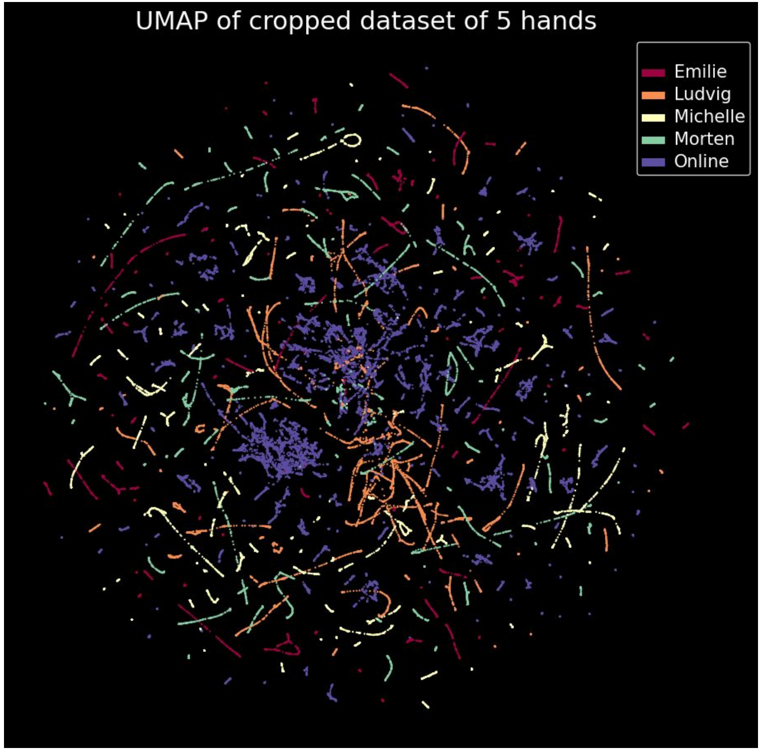
Dimensionality reduction using UMAP on cropped images

Colouring by people (cropped)

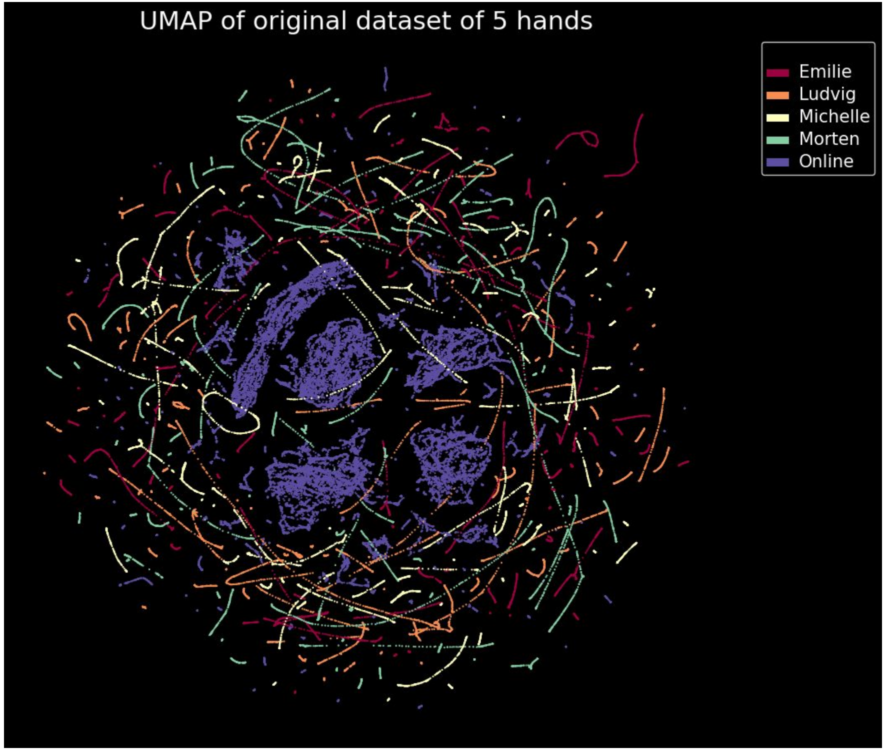


Dimensionality reduction using UMAP on cropped images

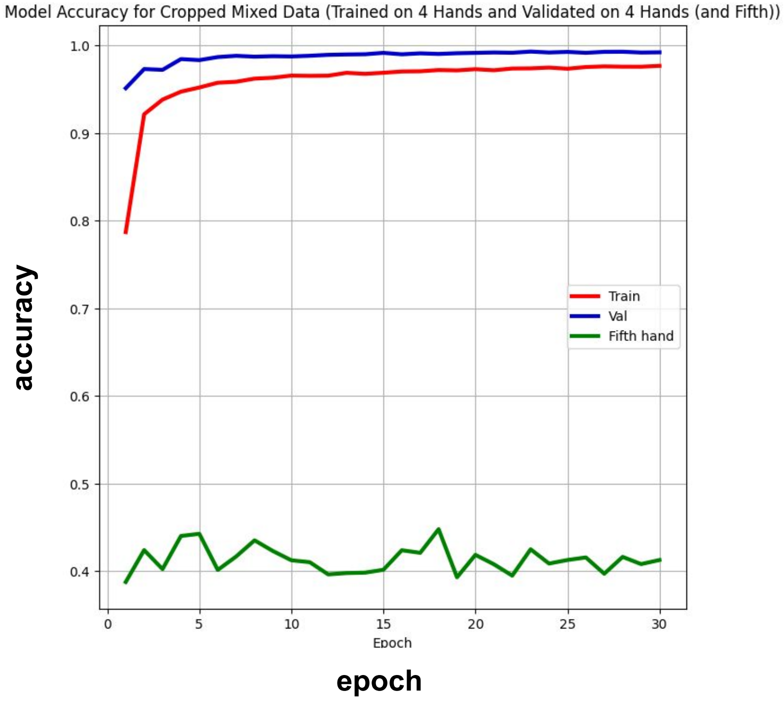
Colouring by people (cropped)

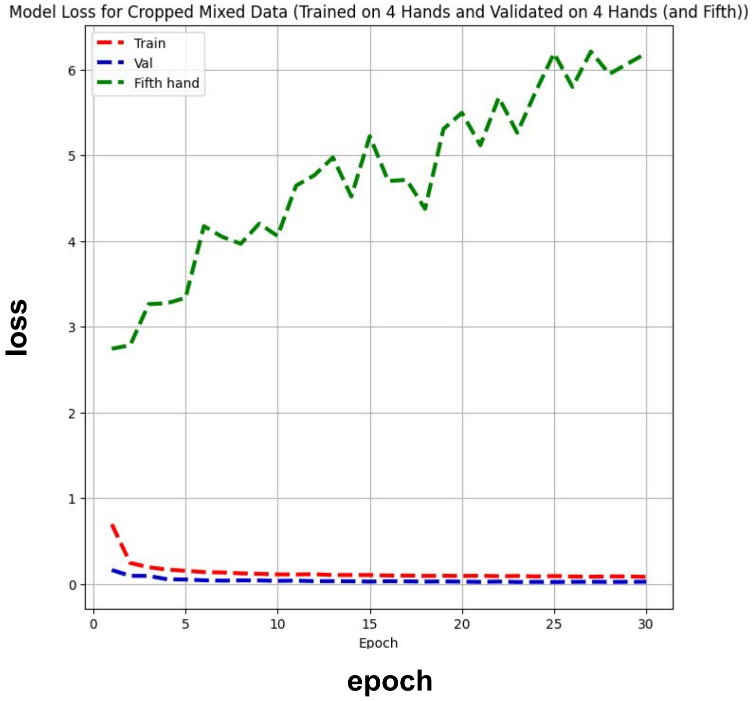


Colouring by people



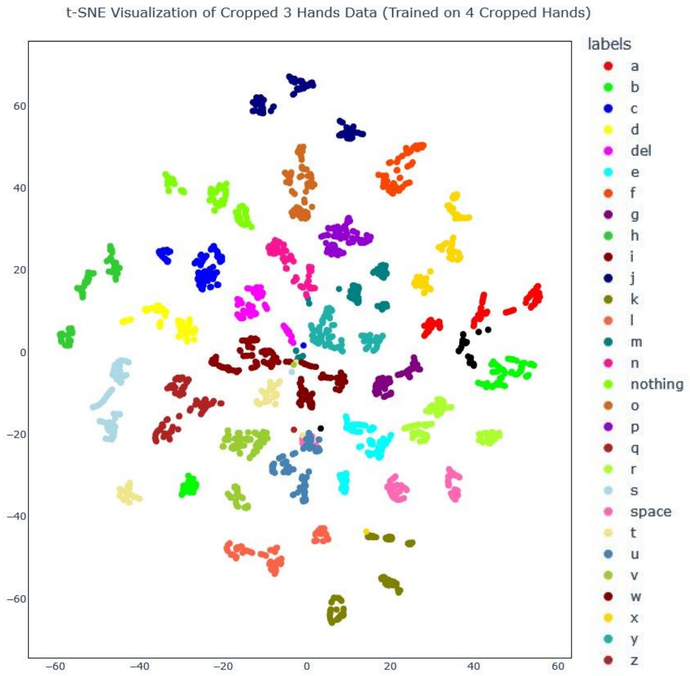
CNN classifies new signs with an accuracy of ~ 40 %





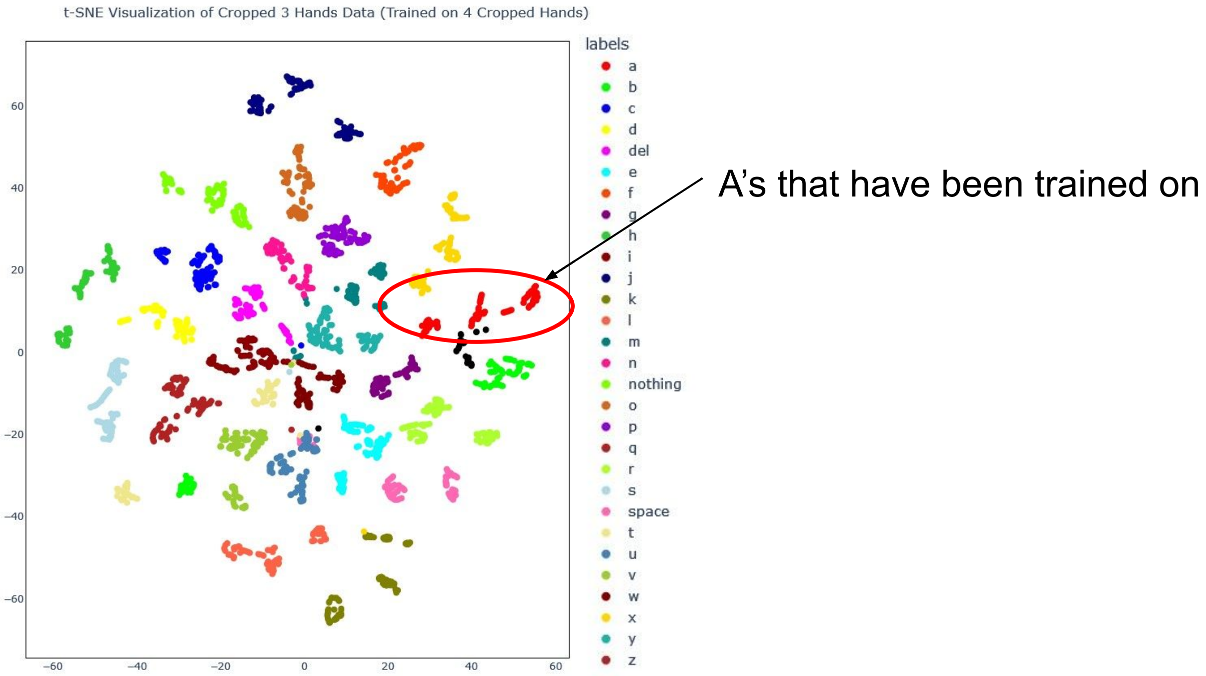
Visualizing Data Using Supervised Clustering

- Is the new hand closer now?



Visualizing Data Using Supervised Clustering

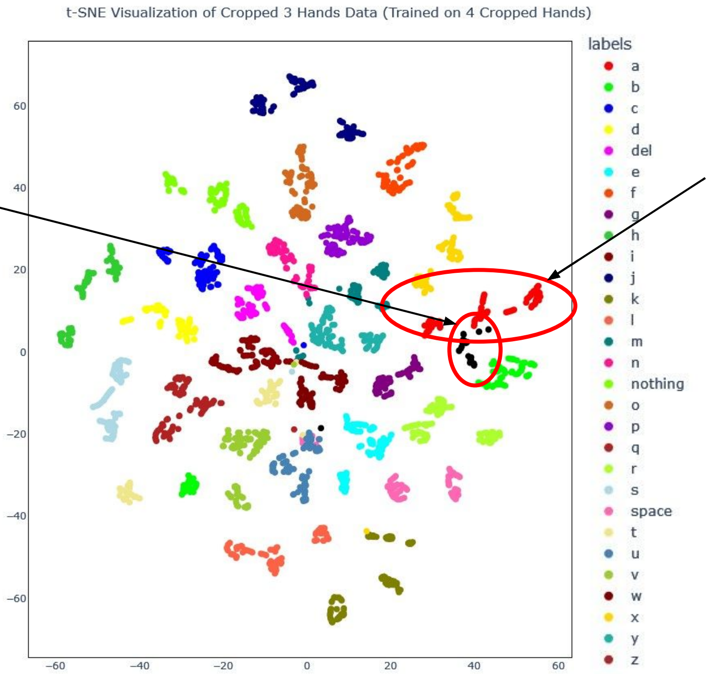
- Is the new hand closer now?



Visualizing Data Using Supervised Clustering

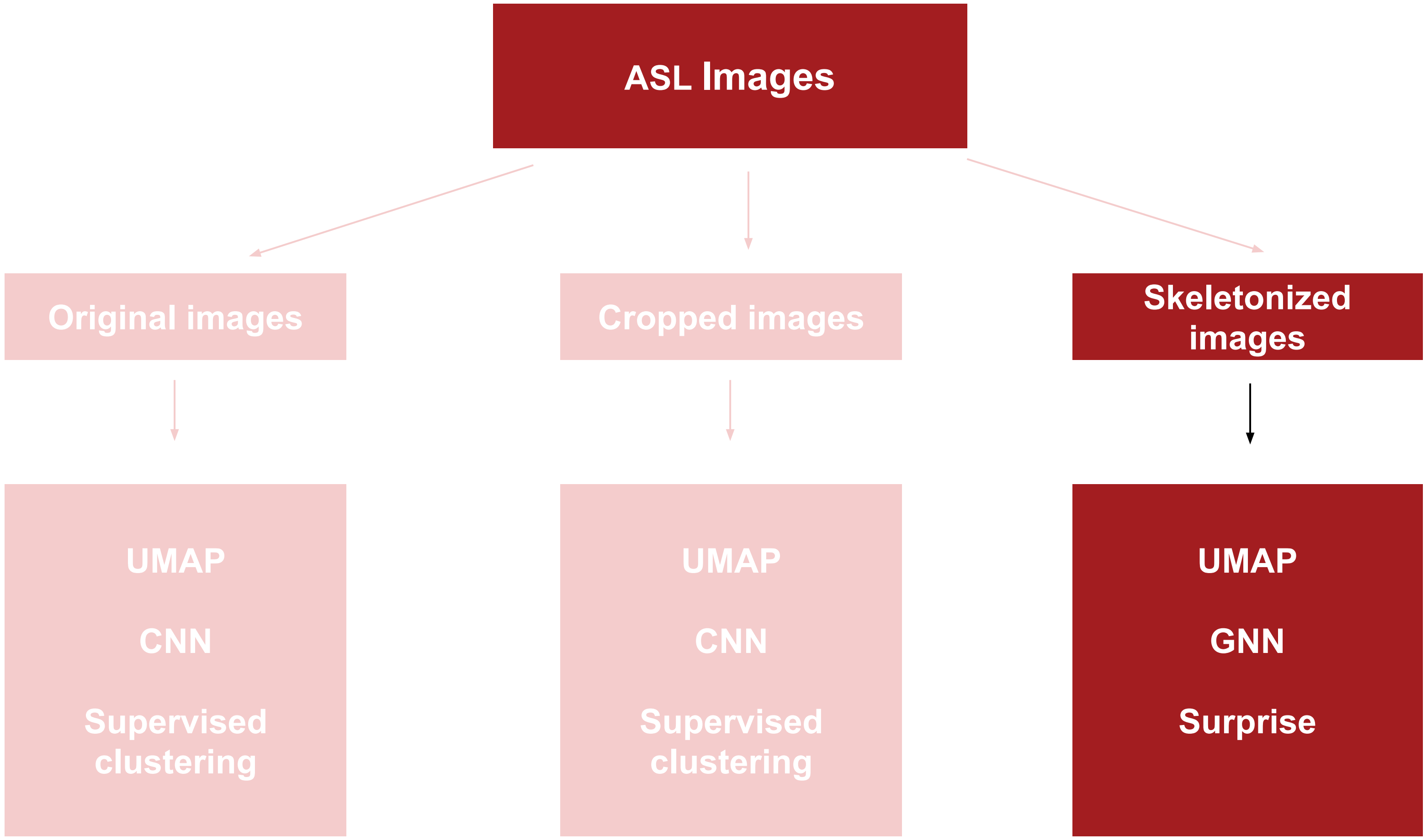
- Is the new hand closer now?

A’s from new hand



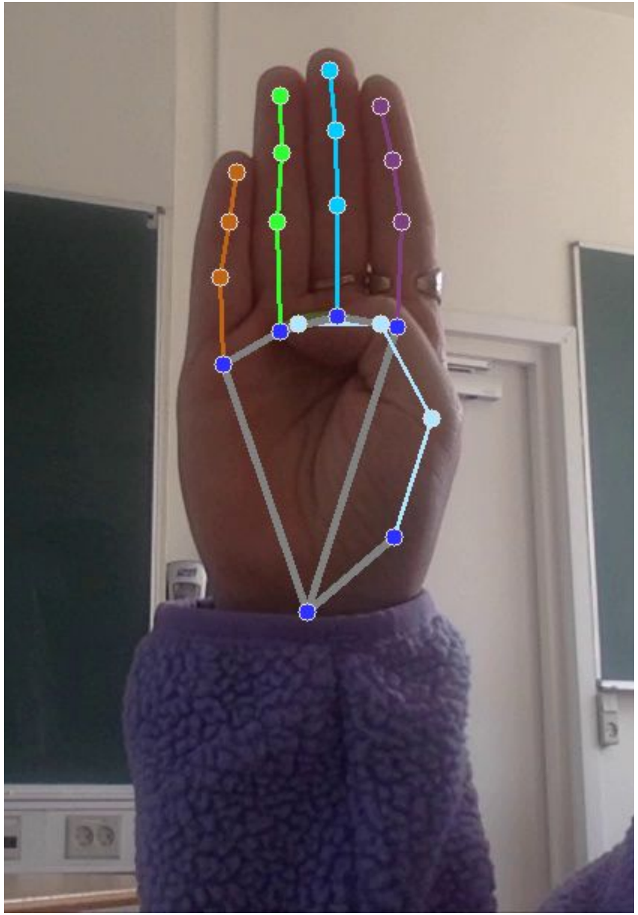
A’s that have been trained on

Conclusion: The network sees less background


ASL Images
Original images
Cropped images
Skeletonized images
UMAP
CNN
Supervised clustering
UMAP
CNN
Supervised clustering
UMAP
GNN
Surprise

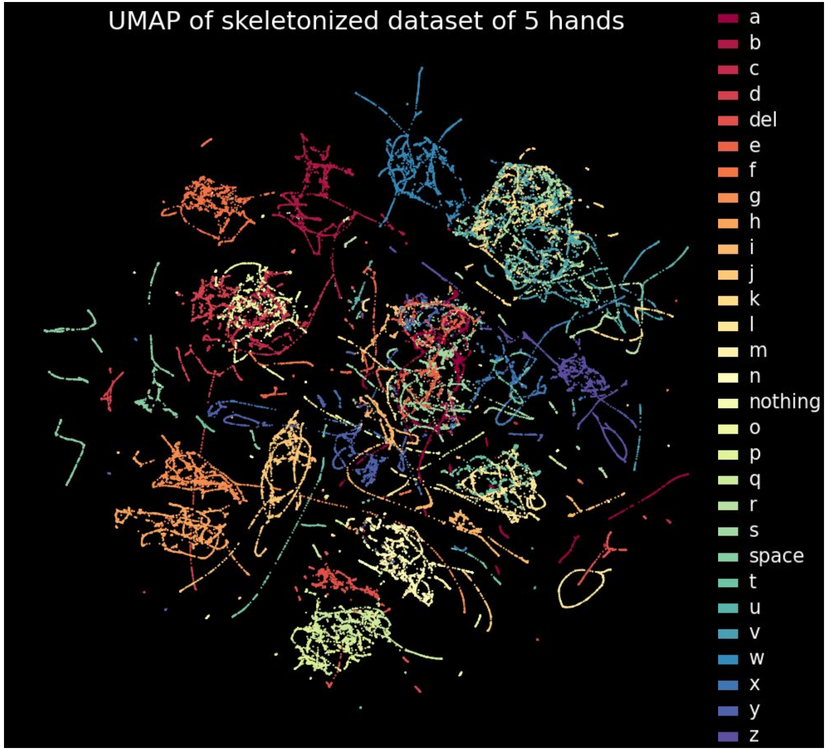
Skeletonized hands

- Use a pretrained CNN to detect landmarks
  - MediaPipe
- Output:
  X, Y and Z coordinates of 21 landmarks
- 100 times smaller than compressed RGB files



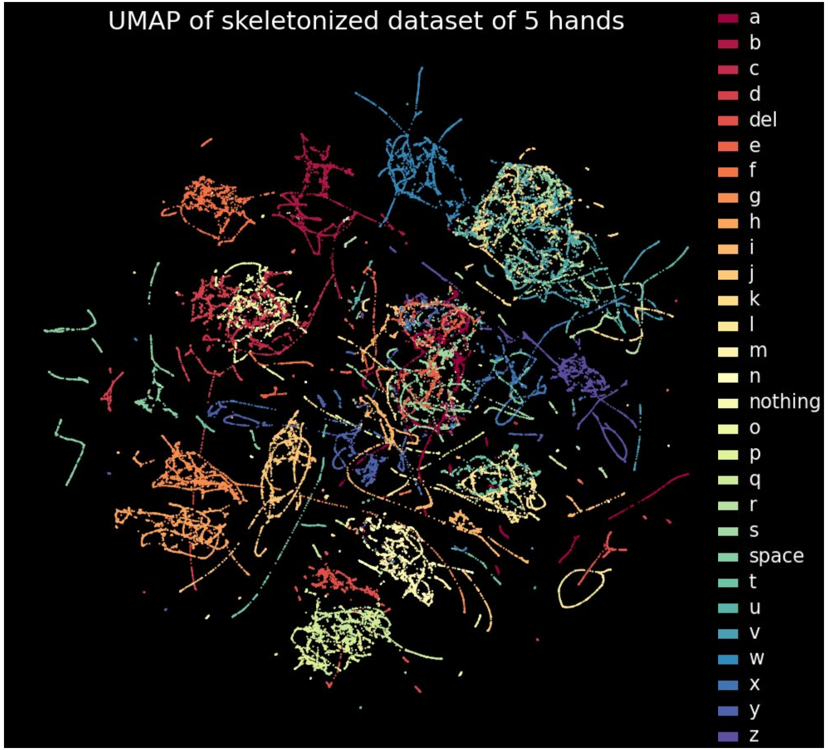
Dimensionality reduction using UMAP on skeletonized images

Colouring by sign (skeletonized)

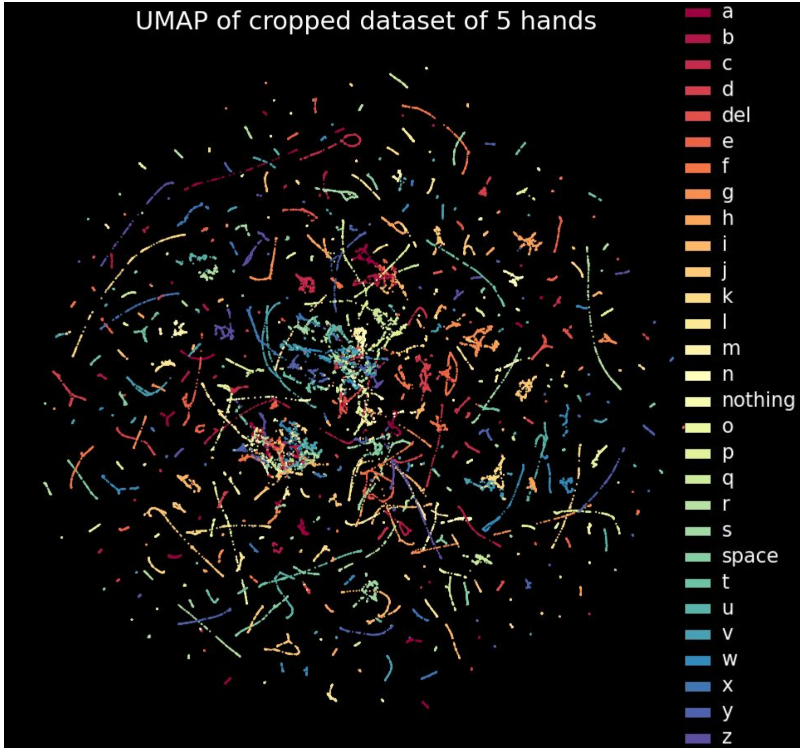


Dimensionality reduction using UMAP on skeletonized images

Colouring by sign (skeletonized)

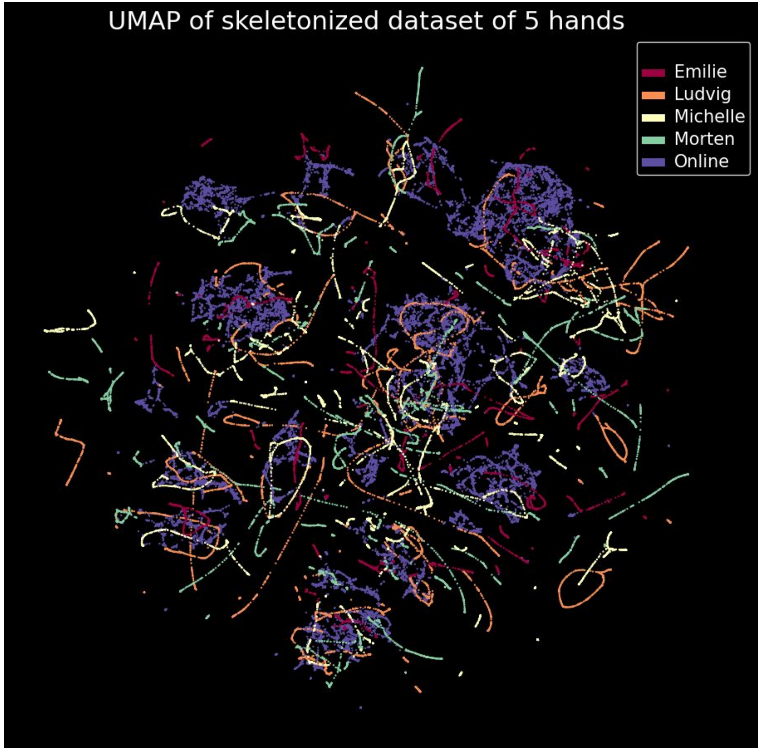


Colouring by sign (cropped)



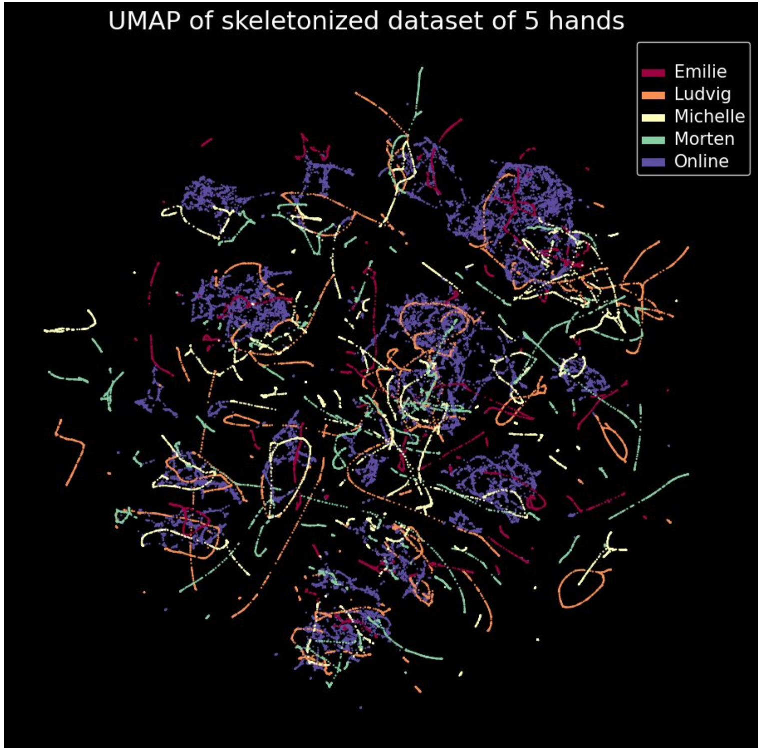
Dimensionality reduction using UMAP on skeletonized images

Colouring by people (skeletonized)

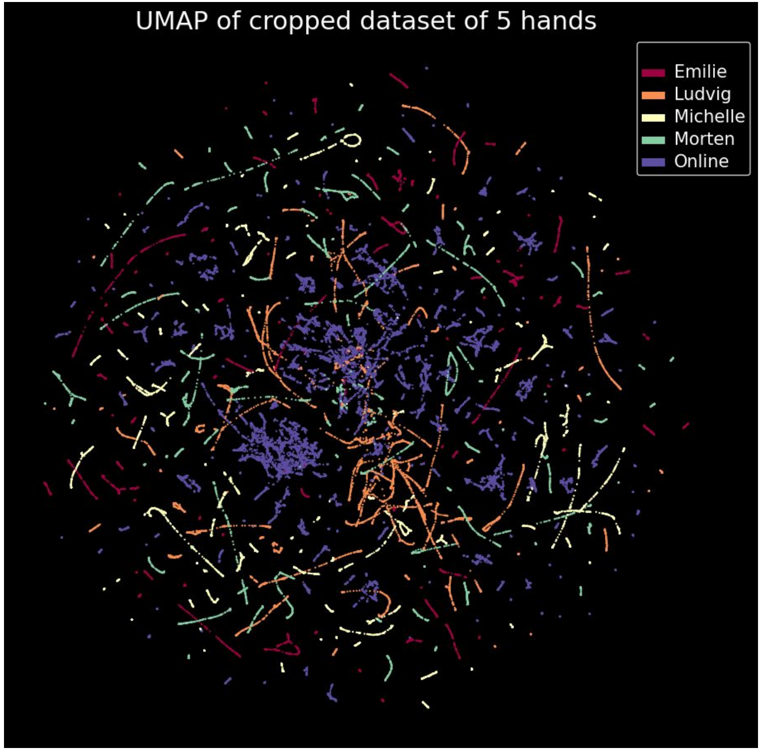


Dimensionality reduction using UMAP on skeletonized images

Colouring by people (skeletonized)



Colouring by people (cropped)

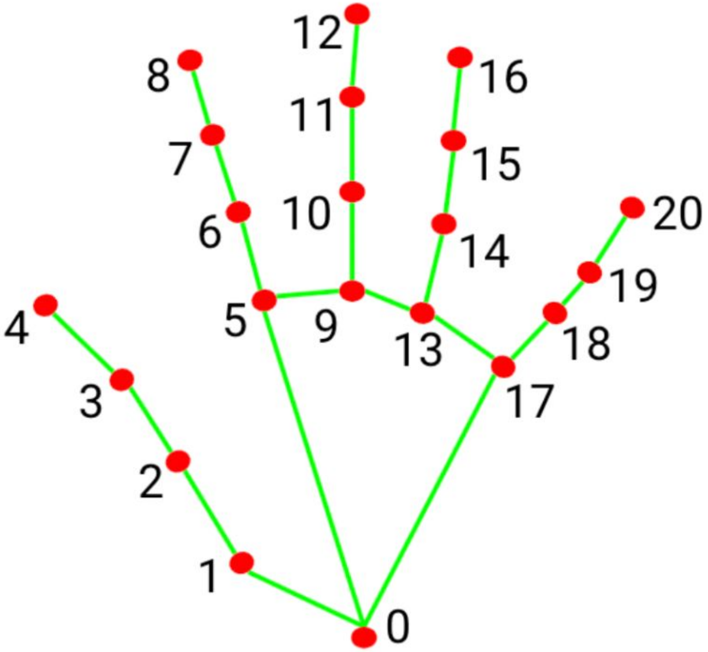


Constructing the graph for a GNN

Each landmark corresponds to 1 node in the graph, with 3 features: X, Y, Z

Basic adjacency matrix:

- Dimensions: 21 x 21
- 1 for connections, otherwise 0



https://developers.google.com/mediapipe/solutions/vision/hand_landmarker#models

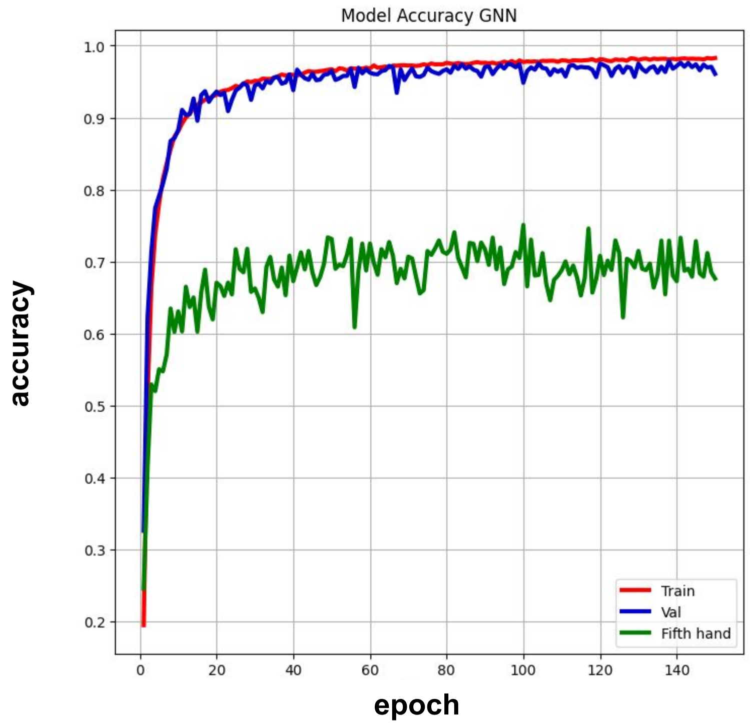
Constructing the GNN for graph classification

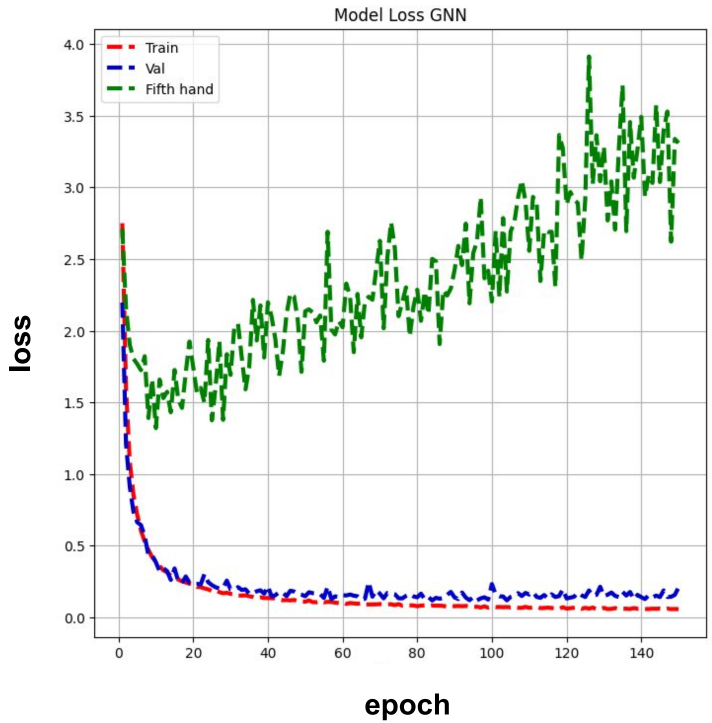
- 1 input GCN layer
- 3 hidden GCN layers
- Pooling layer
- Output layer

Hyperparameters optimized using bayesian search

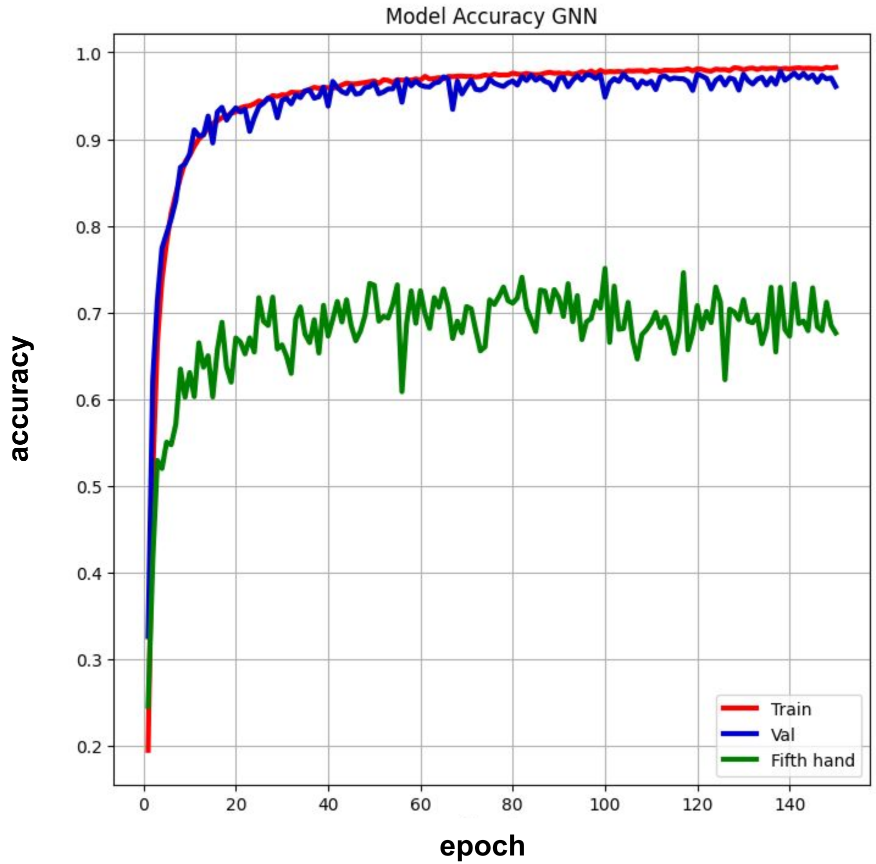
GNN classifies new signs with an accuracy of ~ 70 %

- Adding a new hand in a GNN using the skeletonized data

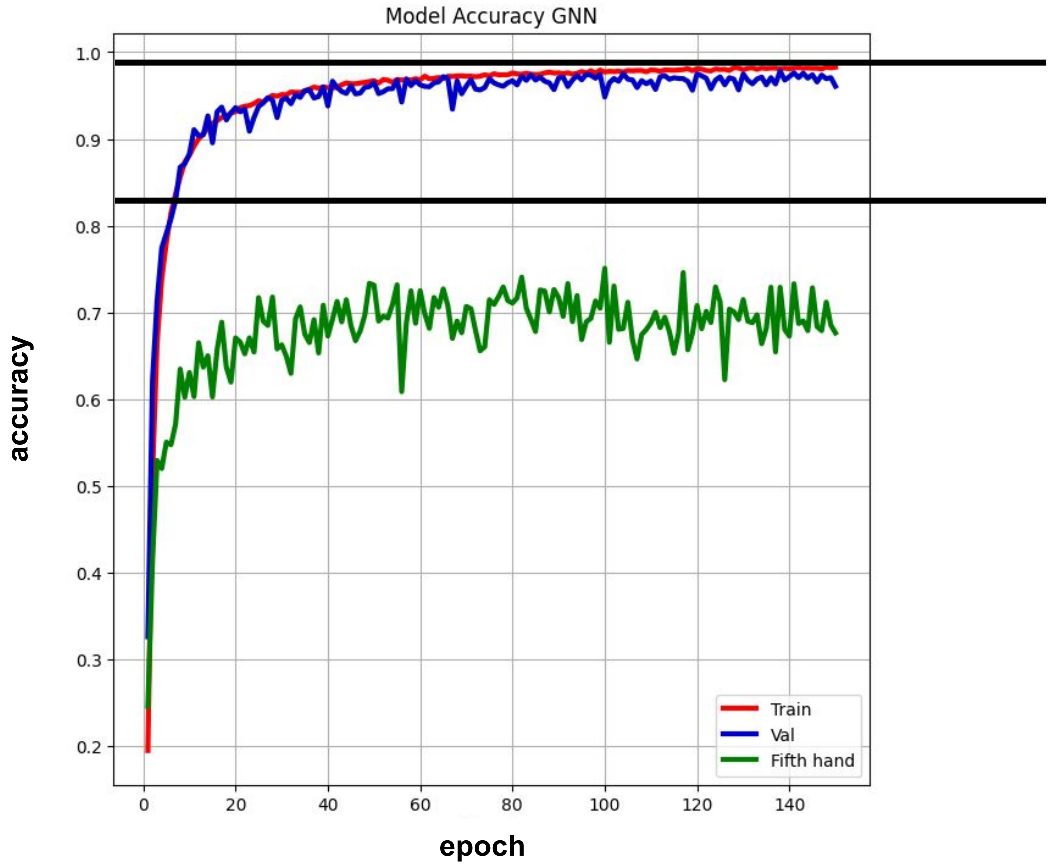




Now we have tabular data…



Surprise, LightGBM is superior

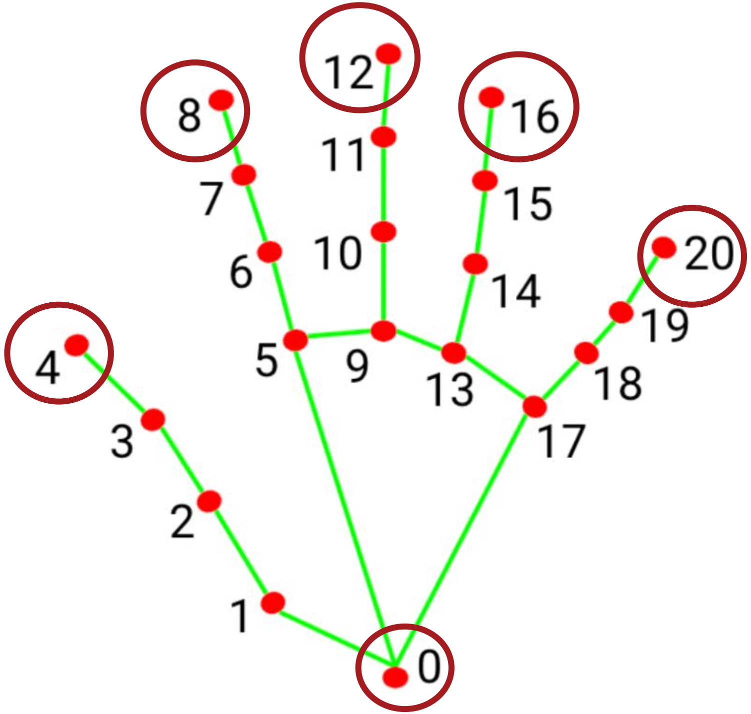


Train: 98.51%

New hand: 83.53%

Top six LightGBM feature selection of landmarks

	feature ranking
0	13610
4	13068
8	12037
20	8511
12	7954
16	7425



https://developers.google.com/mediapipe/solutions/vision/hand_landmarker#models

Conclusion

Preprocessing data and removing background significantly improves CNN performance.

Removing all irrelevant features of images by landmarking dramatically improves accuracy in a GNN.

Once again LightGBM reigns supreme!

Future work

Thoroughly optimize our algorithms

Expanding the training dataset to include a larger variety of hands

Remove failed landmarking using clustering outliers

Appendix

(All participants contributed equally to this project)

Method of reading images

- To make the images from regular images to RGB arrays we have used the library cv2, and the function cv2.imread()

Kaggle images obtained from:
https://www.kaggle.com/datasets/grassknoted/asl-alphabet?datasetId=23079

Hand landmarking

Completed using Mediapipe's HandLandmarker, can be found at

https://storage.googleapis.com/mediapipe-models/hand_landmarker/hand_landmarker/float16/1/hand_landmarker.task

or:

https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

Handedness and real world length estimates are discarded.

Cropping method

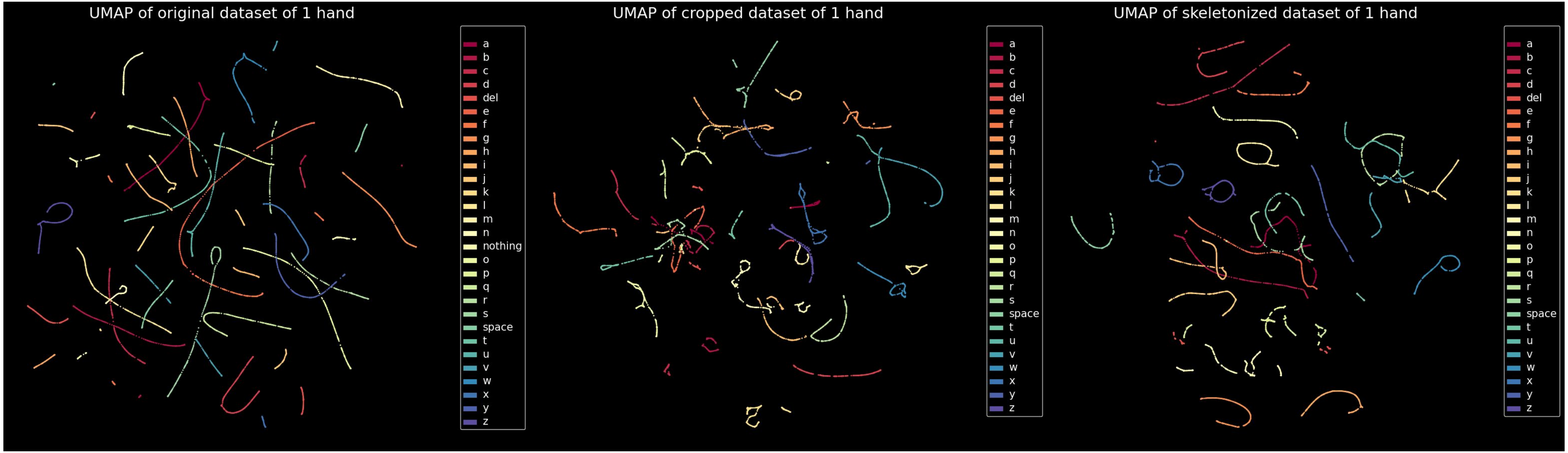
- Follows the procedure for image reading.
- After hand landmarking, the outermost points (xmax, xmin, ymax, ymin) are identified, and a buffer of 0.2*(xmax-xmin) is added to ensure catching the entire hand.

Image compression method

- To make compress the images, we have used the library skimage and the function skimage.transform.resize()

UMAP

UMAP's of only one hand



Performed with umap.UMAP() Settings: default

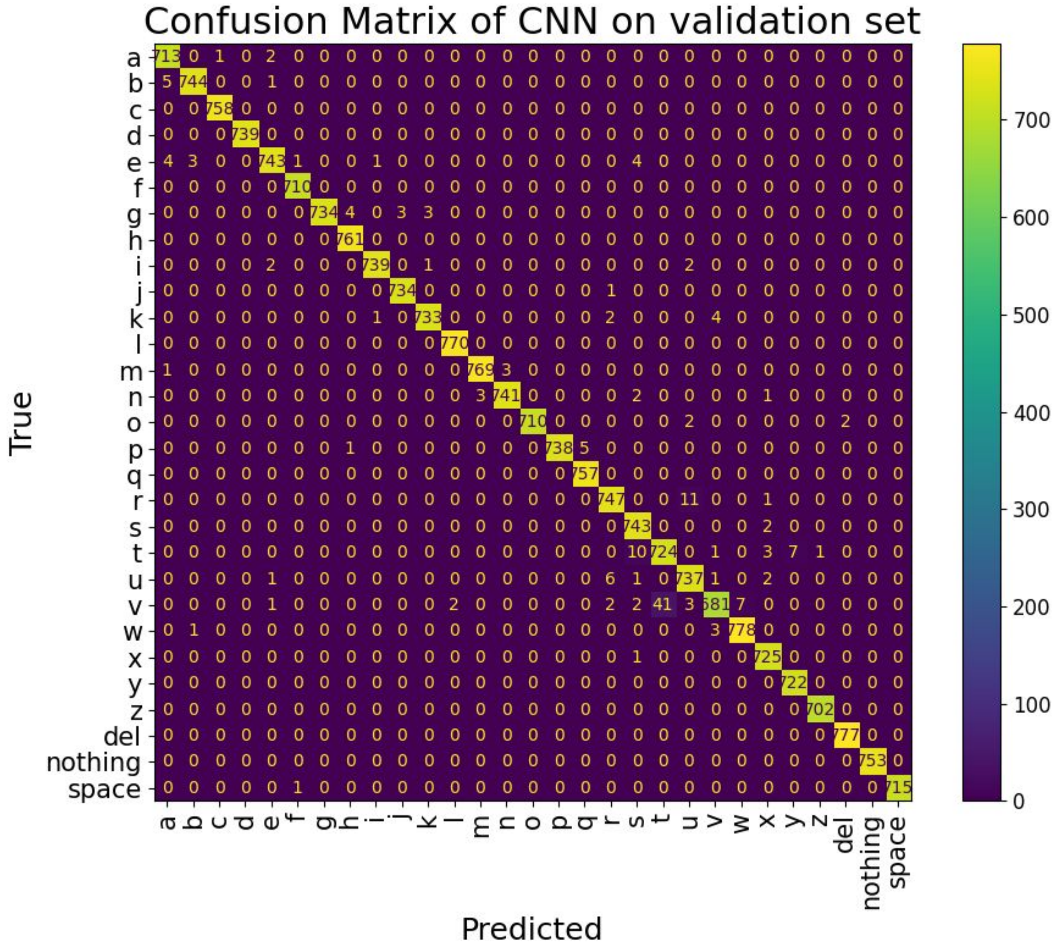
CNN

CNN summary of network in code

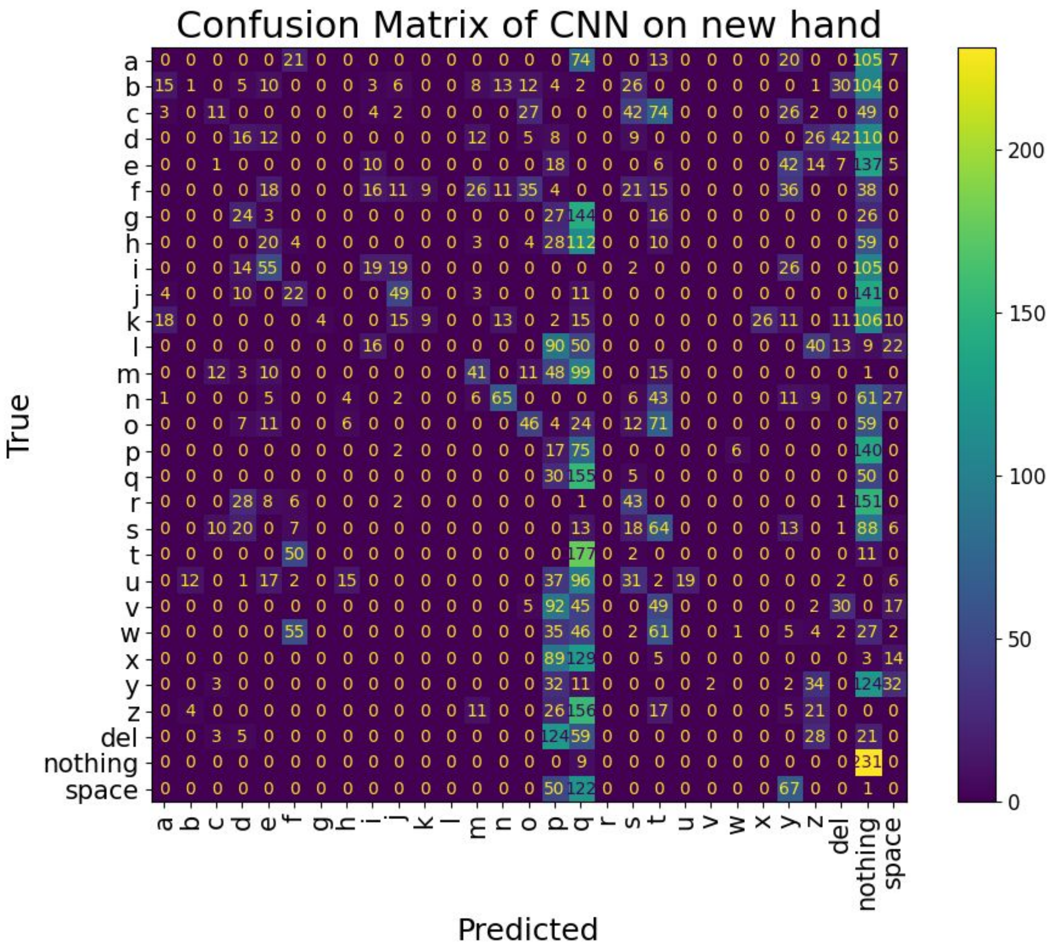
```
base = VGG16(weights = 'imagenet', include_top=False, input_shape=(50, 50, 3))
for layer in base.layers:
    layer.trainable = False

x = base.output
x = Flatten()(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(29, activation='softmax')(x)
```

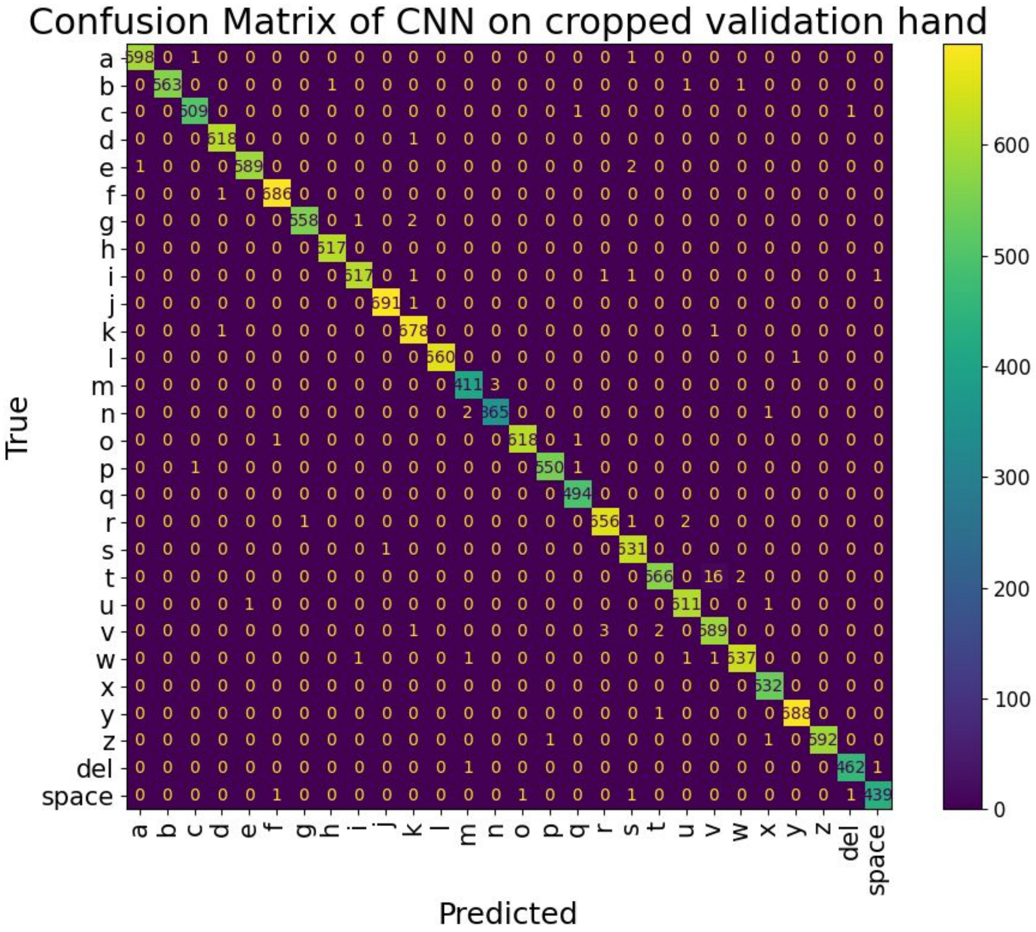
Confusion matrix for CNN on validation set (uncropped)



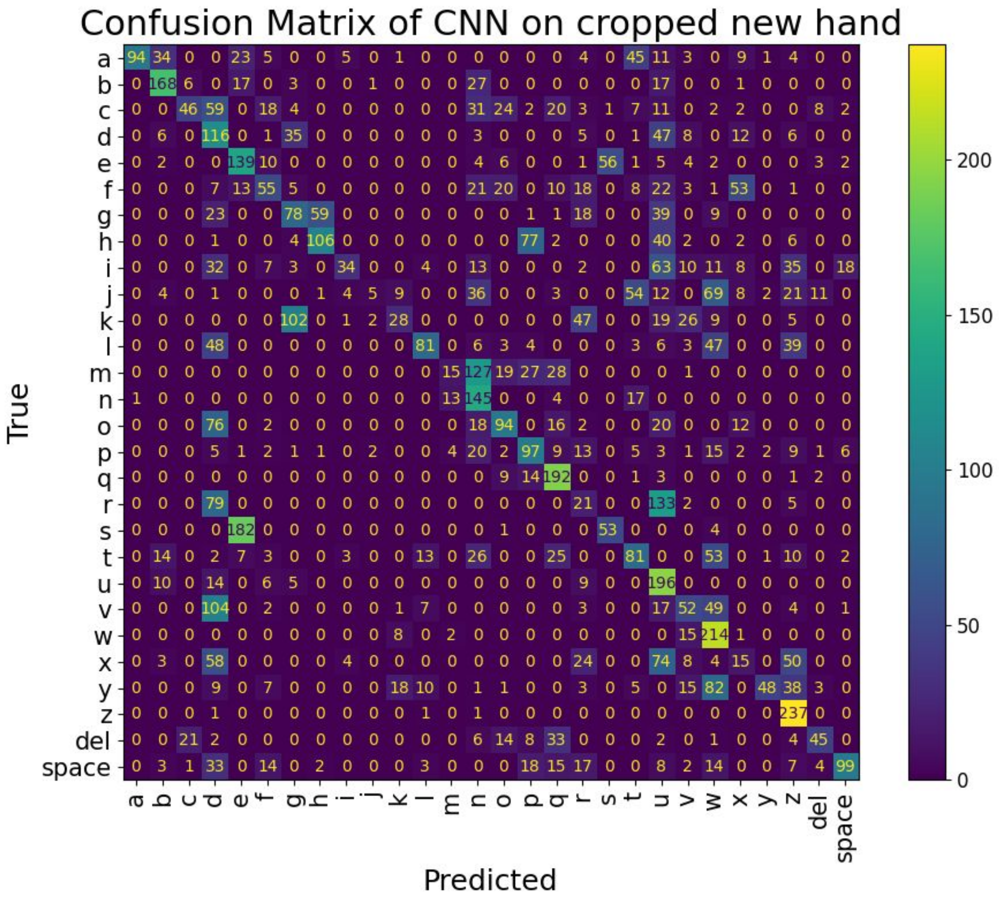
Confusion matrix for CNN on fifth hand (uncropped)



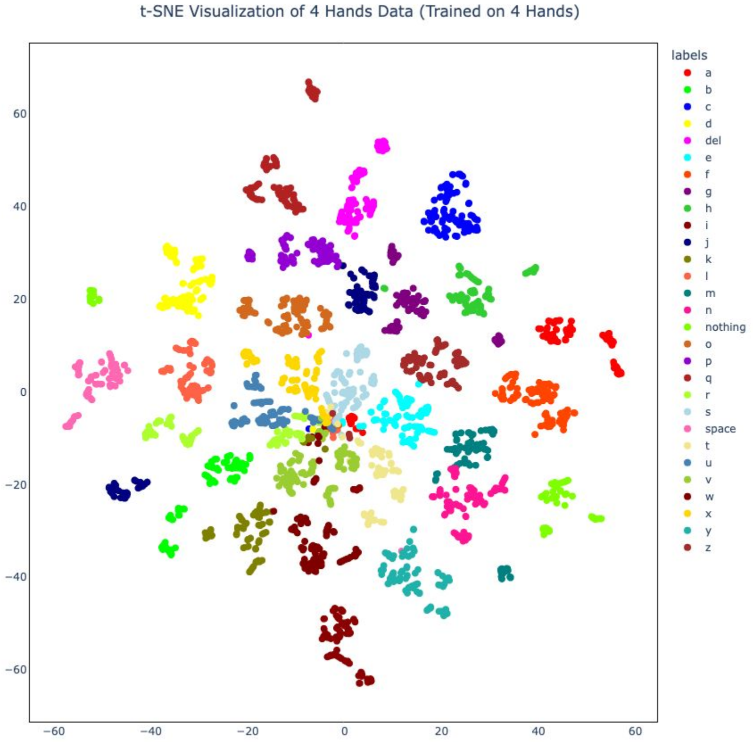
Confusion matrix for CNN on validation set (cropped)



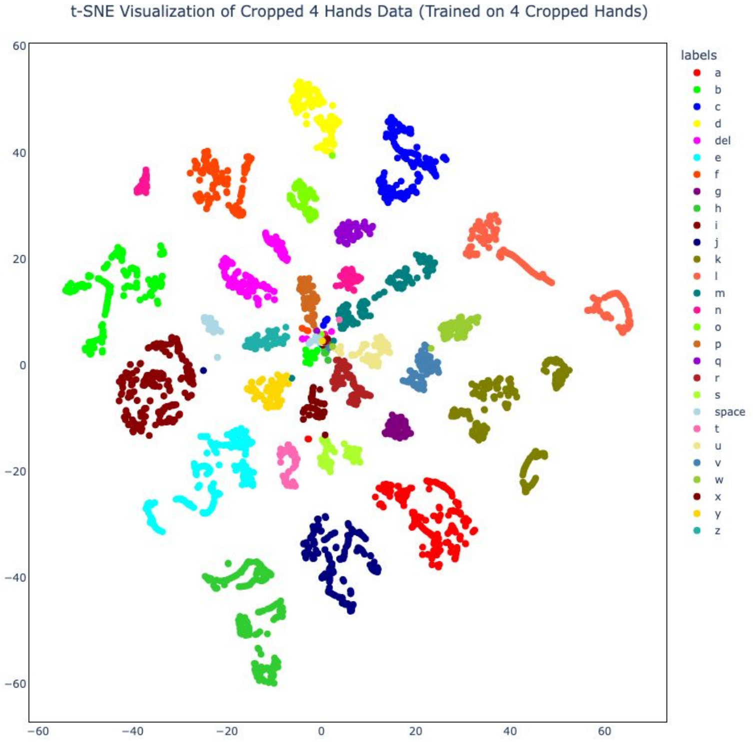
Confusion matrix for CNN on fifth hand (cropped)



t-SNE visualisation on four hands (trained on all four)

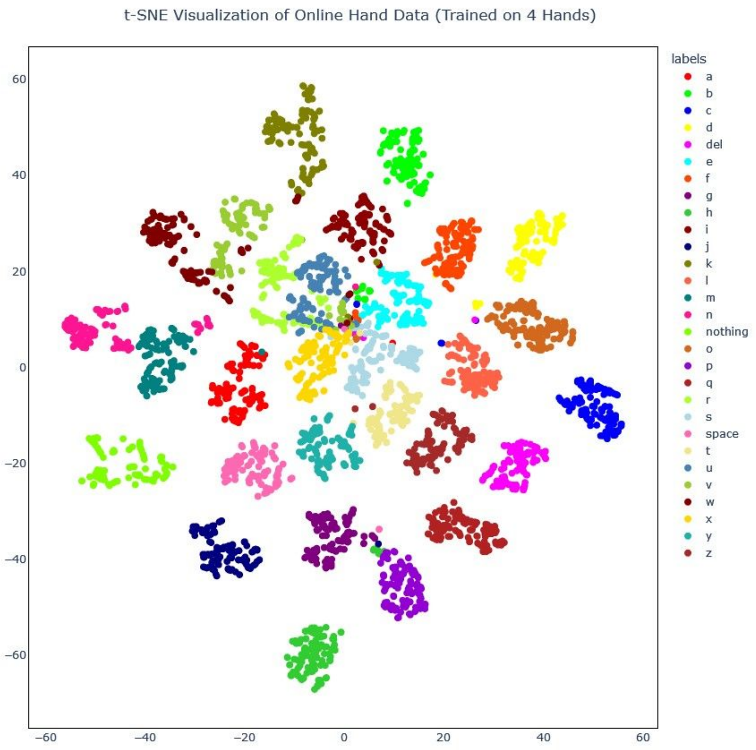


original images

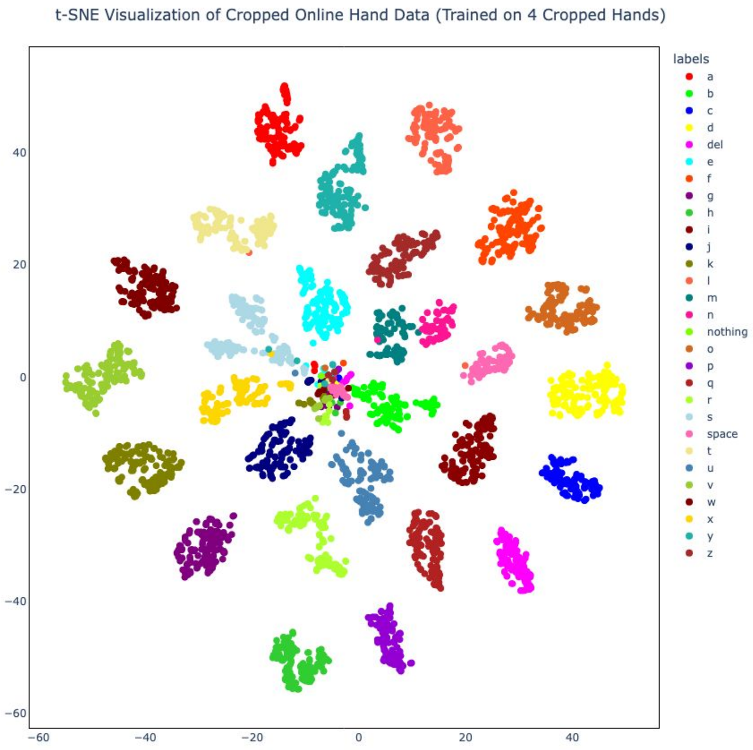


cropped images

t-SNE visualisation of only online hand



original images

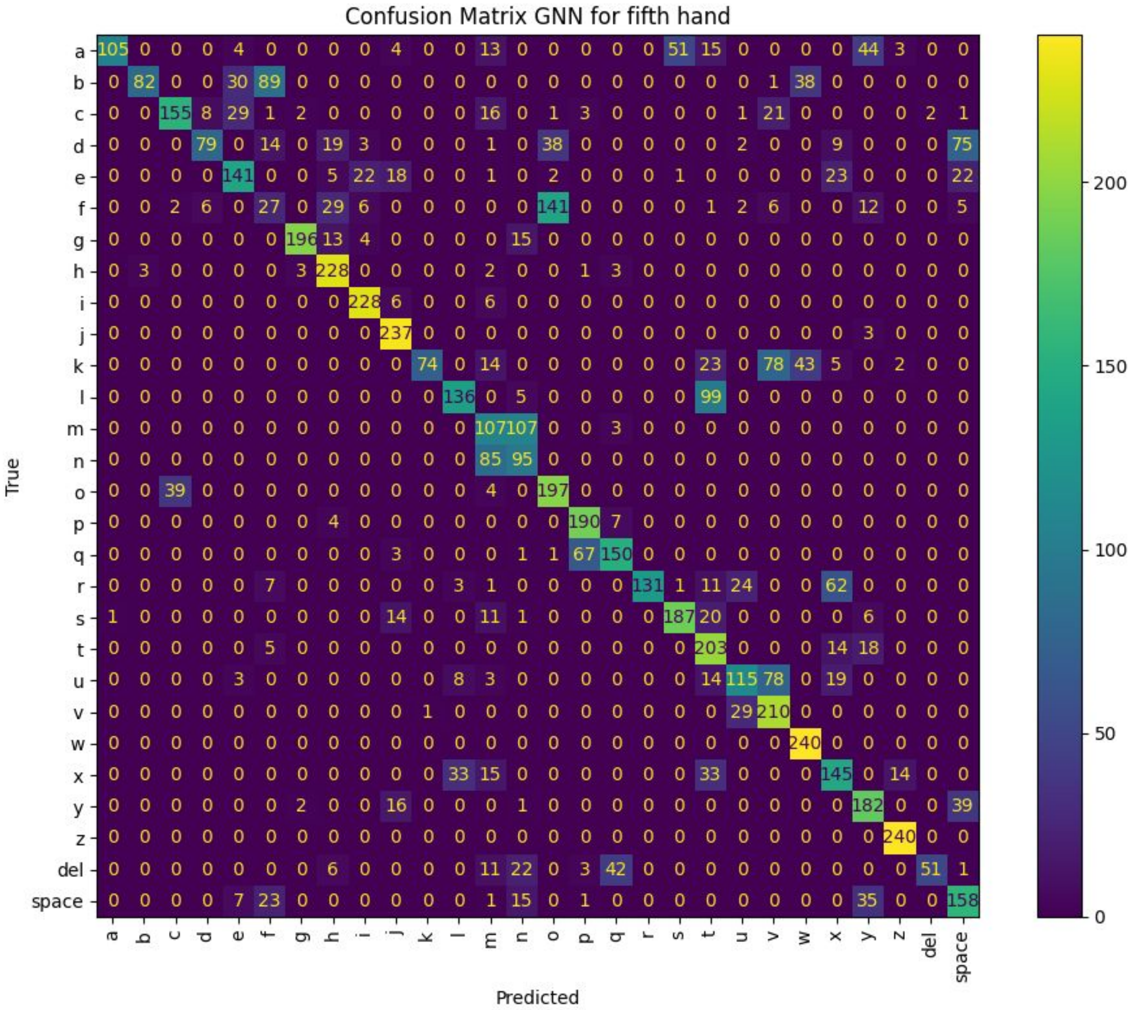


cropped images

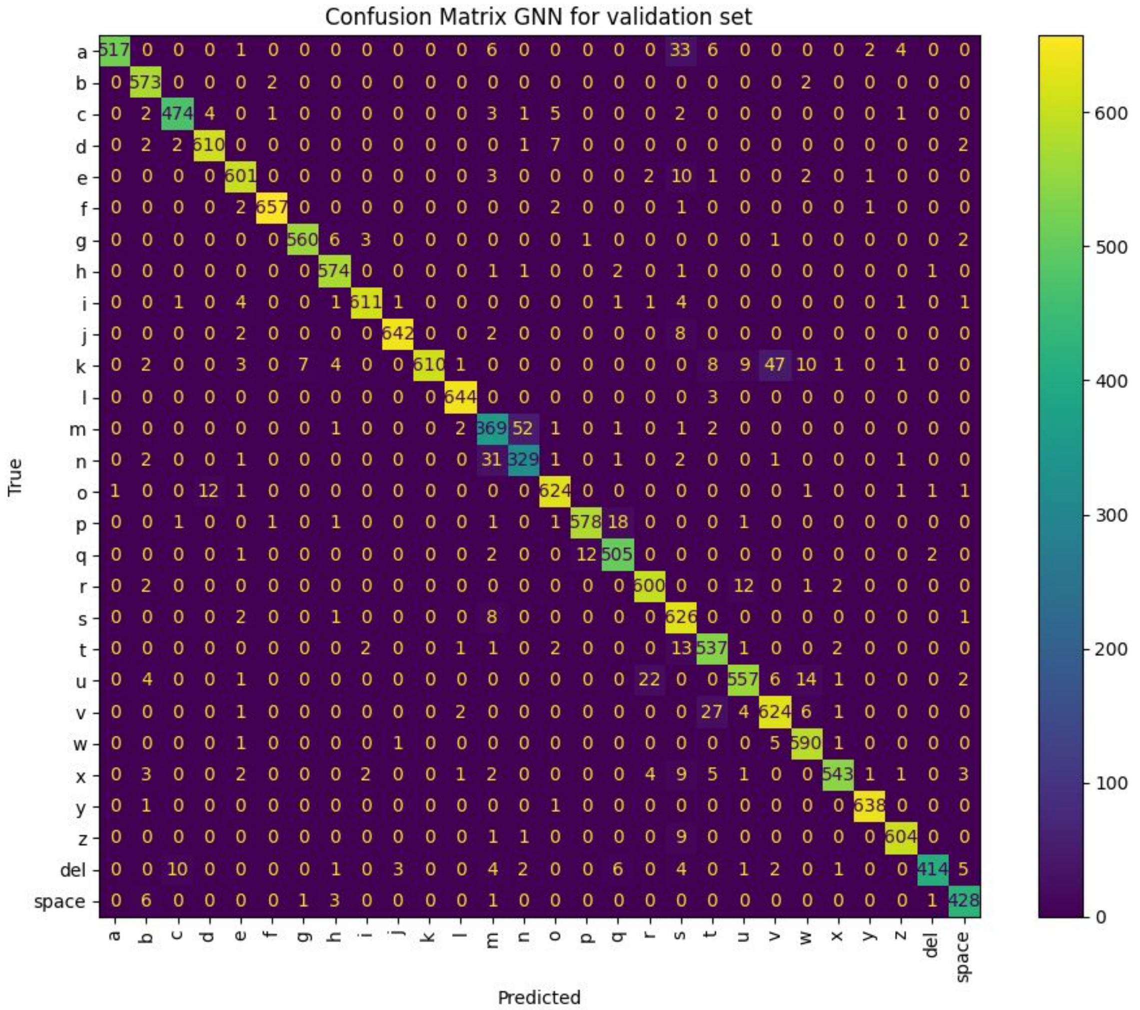
Note: how clusters are now slightly more condensed after cropping

GNN

Confusion matrix for GNN fifth hand



Confusion matrix for GNN Validation set



Graph design

An adjacency matrix was designed based on 3D distance between the nodes, using the simple matrix as edge matrix, but as initial testing was horrible (30%> accurycy on validation) and in the interest of time, this was abandoned.

GNN summary of the network code

```python
class GNNClassifier(tf.keras.Model):
    def __init__(self, hidden_dim, num_classes):
        super(GNNClassifier, self).__init__()
        self.hidden_dim = hidden_dim
        self.num_classes = num_classes

        self.gcn1 = GCNConv(21, activation='relu')
        self.gcn2 = GCNConv(hidden_dim, activation='relu')
        self.gcn3 = GCNConv(hidden_dim, activation='relu')
        self.gcn4 = GCNConv(hidden_dim, activation='relu')
        self.global_pooling = layers.GlobalMaxPooling1D()
        self.fc = layers.Dense(num_classes, activation='softmax')
```

```
Model: "gnn_classifier"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gcn_conv (GCNConv)          multiple                  84

 gcn_conv_1 (GCNConv)        multiple                  2816

 gcn_conv_2 (GCNConv)        multiple                  16512

 gcn_conv_3 (GCNConv)        multiple                  16512

 global_max_pooling1d (Globa  multiple                 0
 lMaxPooling1D)

 dense (Dense)               multiple                  3612

=================================================================
Total params: 39,536
Trainable params: 39,536
Non-trainable params: 0
_________________________________________________________________
```

Graphs are constructed using Spektral

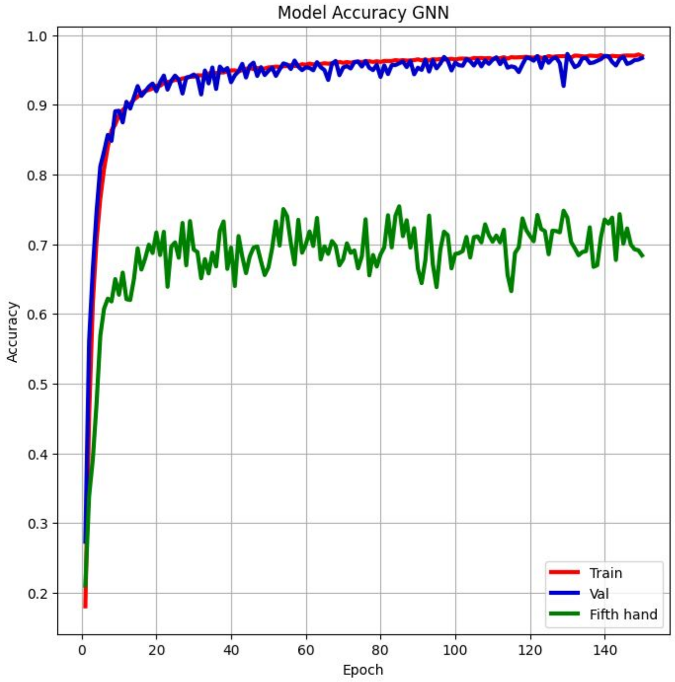
Bayesian search completed using Optuna, optimizing the hidden dimensions, batch size and learning rate.

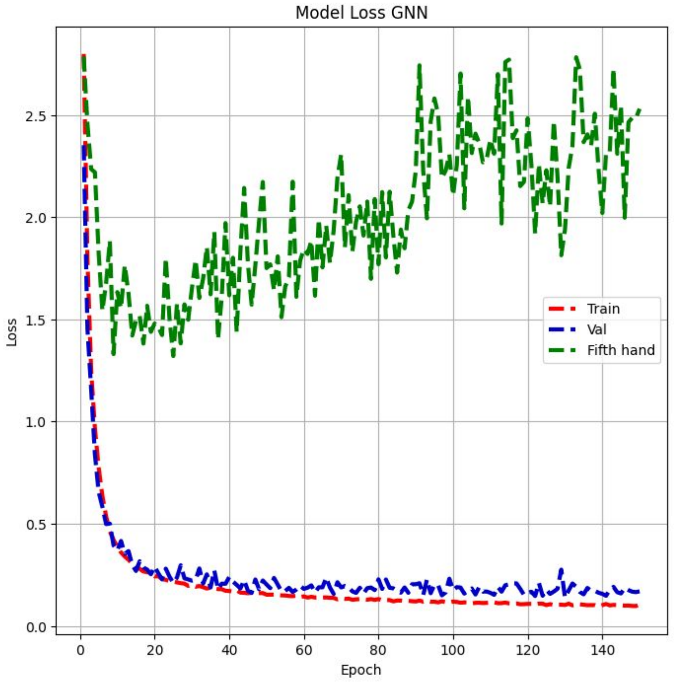
```
hidden_dim = trial.suggest_categorical("hidden_dim", [32, 64, 128])
n_epochs = trial.suggest_categorical("n_epochs", [45])  # Number of epochs
batch_size = trial.suggest_categorical("batch_size", [64, 128, 256, 512, 1024])  # batch size
learning_rate= trial.suggest_categorical("Learning_rate", [0.01, 0.005, 0.0025, 0.001, 0.0005])
```

Optimized using Adam. loss function: Keras.losses.SparseCategoricalCrossentropy

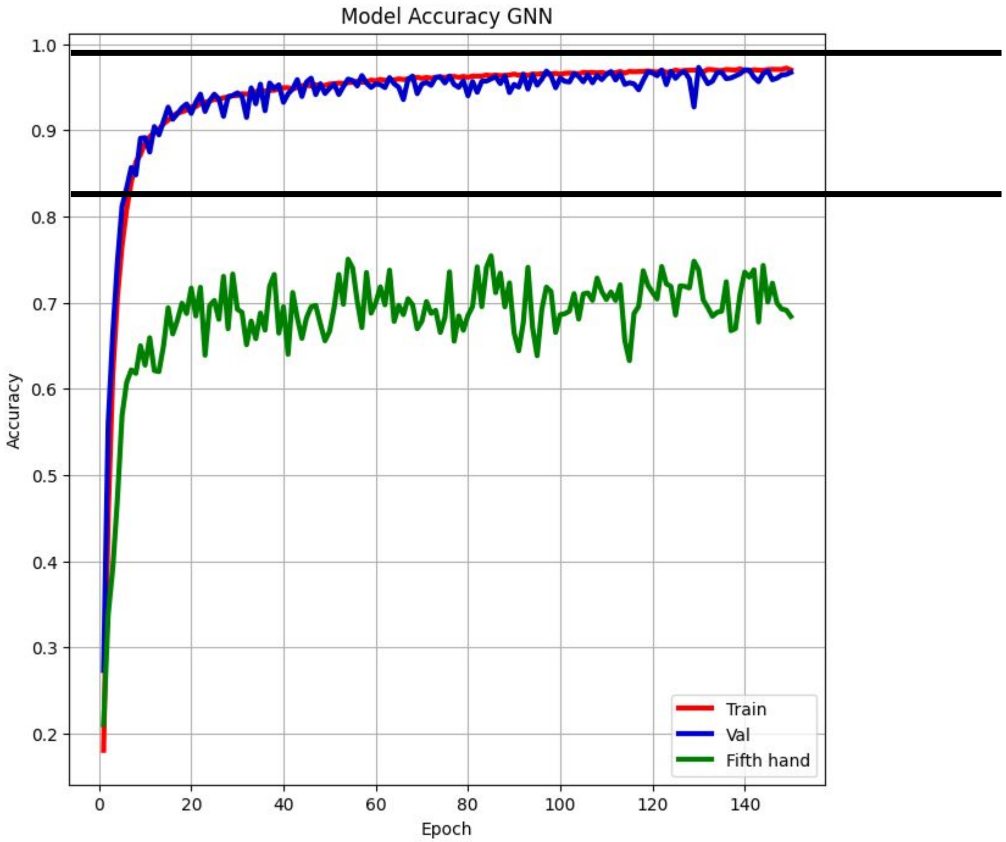
GNN classifies new signs with an accuracy of ~ 70 % pre optimisation

- Adding a new hand in a GNN using the skeletonized data





LightGBM comparison to GNN pre optimization



Validation: 98.51%

New hand: 83.53%

LightGBM

LightGBM hyperparameters changed from default

- objective = ‘multiclass’
- num_leaves = 35
- n_estimators = 125

Feature importance for each coordinate LightGBM using built-in ranking

index	feature	ranking
0	Y4	6467
1	Y8	5599
2	Z0	5298
3	X4	5006
4	X8	4977
5	X0	4746
6	Y20	4279
7	Y16	3961
8	Y0	3566
9	Y12	3490
10	X12	3200
11	X20	2611
12	Z5	2488
13	Z1	2426
14	Y1	2004
15	X1	1931
16	Z17	1862
17	Z16	1782
18	X16	1682
19	Y3	1647
20	Z20	1621
21	Y2	1607
22	Z4	1595
23	X2	1588
24	Y17	1577
25	Y18	1554
26	X5	1547
27	Y5	1479
28	Z8	1461
29	X6	1457
30	Y10	1444
31	Y14	1397
32	Y15	1393
33	Y6	1387
34	X17	1296
35	Z12	1264
36	Z14	1245
37	Y11	1200
38	Z9	1185
39	X3	1151
40	Z18	1072
41	X7	1047
42	Y7	1042
43	Y19	1040
44	Z13	996
45	Z3	912
46	Y9	856
47	Z2	854
48	Z11	827
49	X11	810
50	X10	793
51	Z7	789
52	X19	742
53	Z10	730
54	Z15	715
55	X18	712
56	Y13	677
57	Z19	664
58	X9	655
59	Z6	627
60	X14	542
61	X15	512
62	X13	495

Feature importance for each landmark LightGBM using built-in ranking

feature	ranking
0	13610
4	13068
8	12037
20	8511
12	7954
16	7425
1	6361

5	5514
17	4735
2	4049
3	3710
6	3471
18	3338
14	3184

10	2967
7	2878
11	2837
9	2696
15	2620
19	2446
13	2168

Confusion matrix by LightGBM