ARABIC SIGN LANGUAGE RECOGNITION USING A NEURAL NETWORK

Sara Ali Al Messabi - 202215001
Aysha Alneyadi 202102162
Alreem Alkaabi - 202307361
Sarah Alyammahi - 202200542
Alyazia Khalifa Alblooshi 202212335

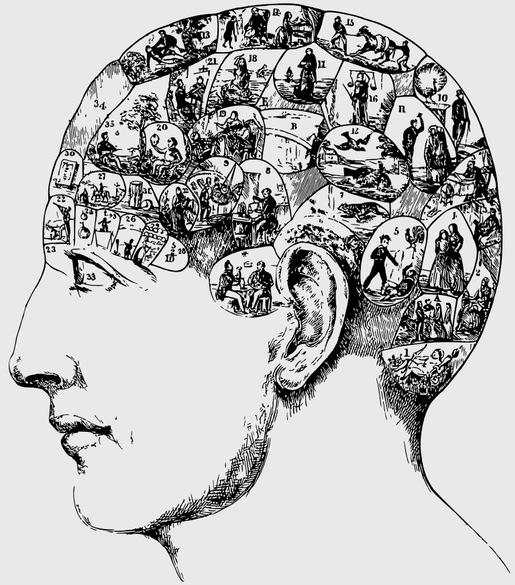


TABLE OF CONTENT :

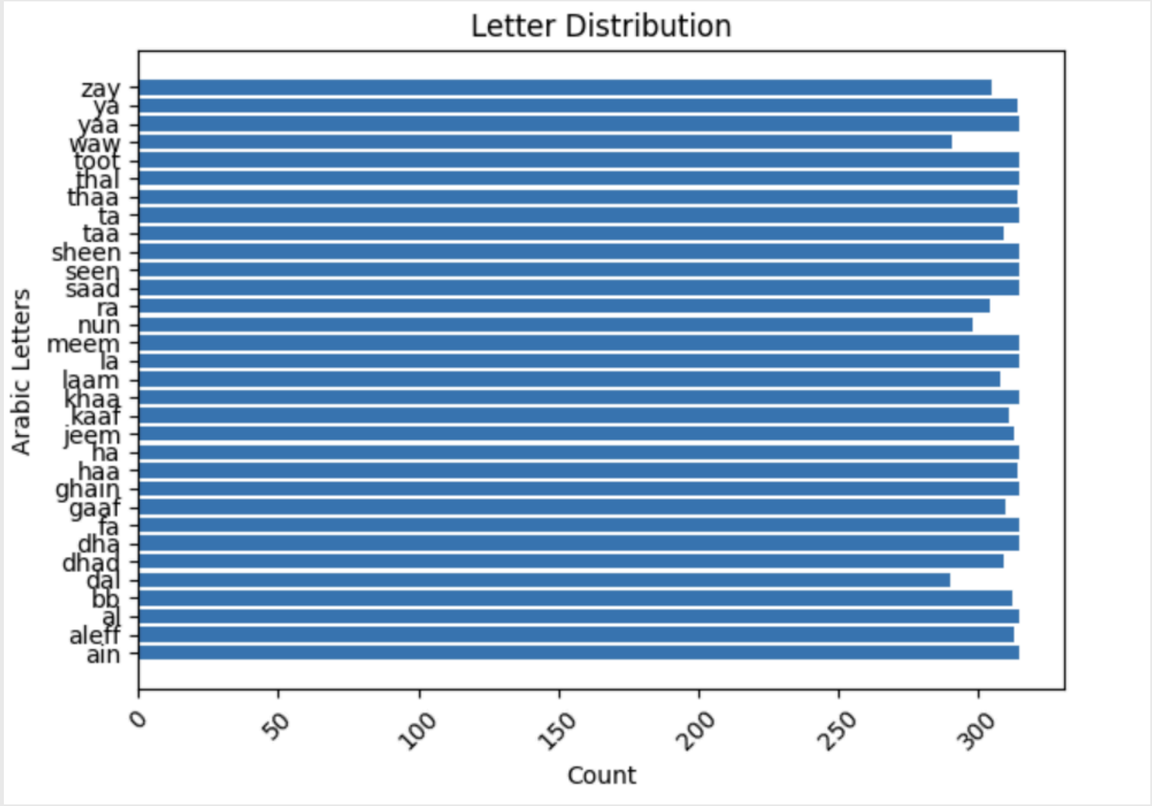
The Goal:

- Our objective is to train a CNN on the Arabic sign language

أ ب ه

Understanding The Dataset:

- The Dataset has 32 Classes
- It contains a training file with 9k images
- It also contains a validation file with 4k images that we used to validate our model
- The dataset is balanced as shown in the graph



Preprocessing :

Step 1: File Format

Before applying augmentation we realized the images in the file are in the wrong format for training and validation. We applied the code below to create a file for each class and put the images there. This also helped when we needed to use pre-trained model since it requires this file format.

To work with most pre-trained model and pytorch it requires the dataset to be orgainzed in a certain format inside the file, this code changes the file format to fit it.

```
import os
import shutil
#Loading the training image files with label
train_image_file_dir=r"C:\\Users\\saraa\\Desktop\\datasetRNET\\train\\images"

#Loading the validation images files with label
valid_image_file_dir=r"C:\\Users\\saraa\\Desktop\\datasetRNET\\valid\\images"
name_to_id = {v:int(k) for k,v in class_name.items()}

def organize_file(dr):
    for file in os.listdir(dr):
        class_id=file.split("_")[3]
        if class_id in name_to_id:
            class_folder = os.path.join(dr, class_id)
            if not os.path.exists(class_folder):
                os.makedirs(class_folder)
            shutil.move(os.path.join(dr,file),os.path.join(class_folder,file))
organize_file(train_image_file_dir)
```

Preprocessing :

Step 2 : Augmentation

We used:

1.Resize Images to a Fixed Size of (224,224) since it is required by most pre-trained models

2.Random Horizontal Flip, good for sign language since people are left and right handed.

3.Random Rotate, important since sign language can appear in many angles.

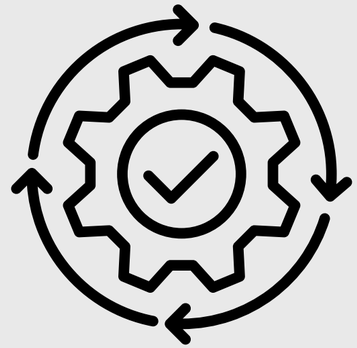
4.Color Jitter, for bad image quality.

4.Transforming it to Tensor format for training.

5.Normalizing the images.

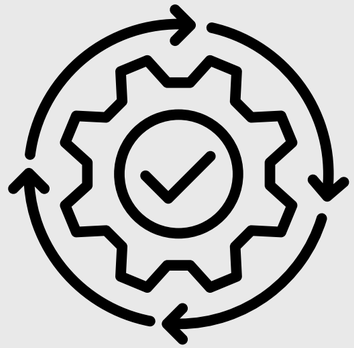
Training will have a batch_size=16

Models Used:



1. **A Customized CNN**
2. **Resnet50**
3. **MobileNetV3**

First Model: Customized CNN



- CNN automatically learns features like shapes and edges from sign images.
- Our model uses multiple Conv2D, Batch normalization, and Dropout layers to enable the model to learn complex patterns while preventing overfitting.
- We chose a custom built CNN first to test our knowledge and attempt to build a CNN that fits better than pre-tained models with Arabic sign language.

Architecture of the Model and Challenges

- Large Architecture require smaller batch size since our devices don’t have a lot of memory
- This leads it to needing a lot of time that we didn’t have on hand
- Overfitting was one of our major problems in this project

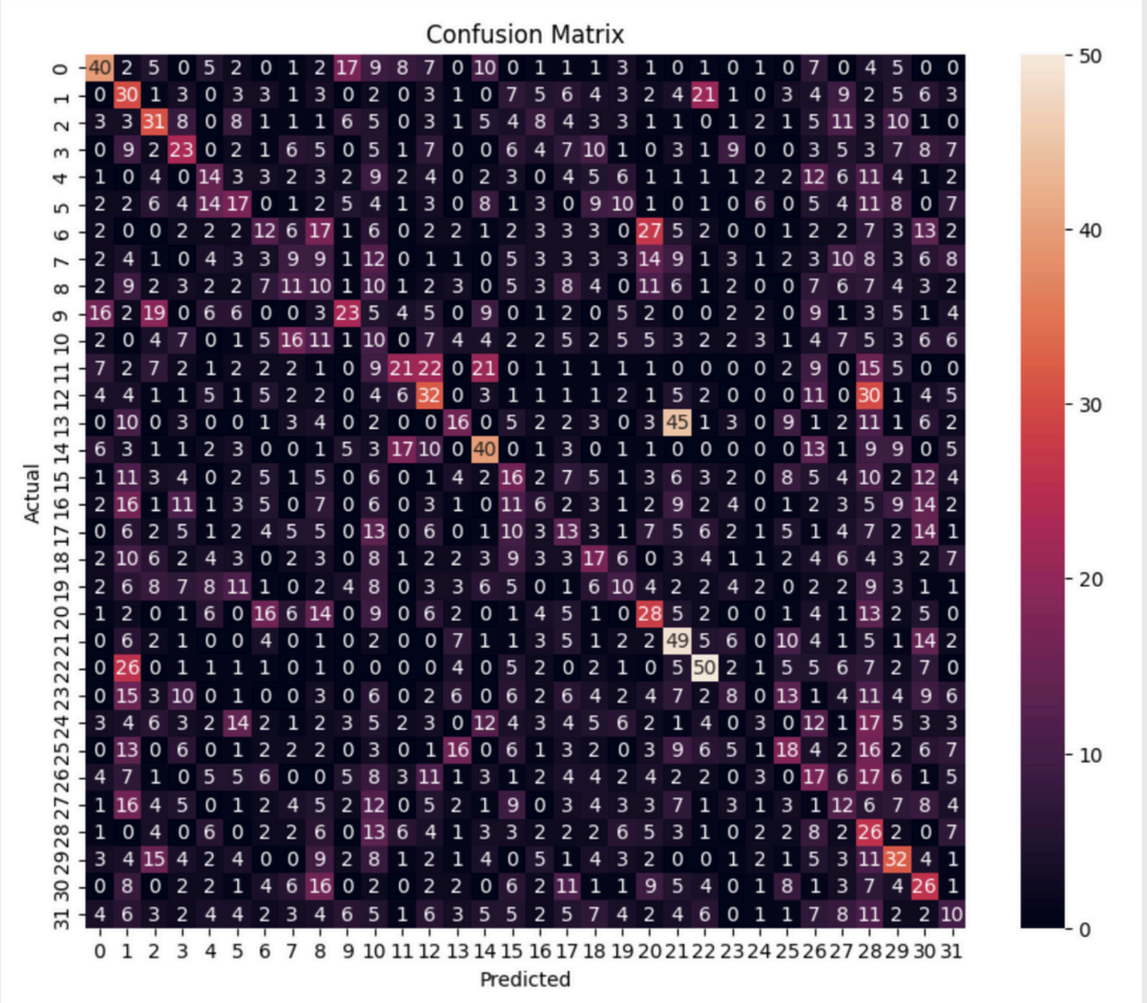
```
CNN(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1))
  (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1))
  (bn3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (dropout): Dropout(p=0.4, inplace=False)
  (fc1): Linear(in_features=86528, out_features=32, bias=True)
)
```

Result of the Model:

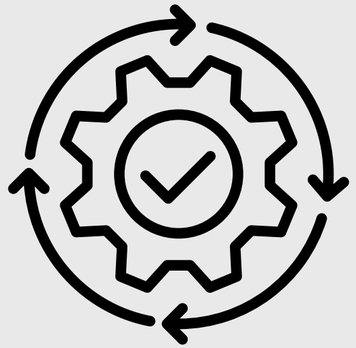
```
The training dataset Accuracy:
Got 7083/9955 with accuracy 71.15%
The Validation dataset Accuracy:
Got 669/4247 with accuracy 15.75%
```

Classification Report:

	precision	recall	f1-score	support
ain	0.36	0.30	0.33	133
al	0.13	0.22	0.16	135
aleff	0.22	0.23	0.22	134
bb	0.19	0.17	0.18	135
dal	0.14	0.13	0.13	111
dha	0.16	0.13	0.14	135
dhad	0.12	0.09	0.10	130
fa	0.10	0.07	0.08	135
gaaf	0.06	0.07	0.07	134
ghain	0.27	0.17	0.21	135
ha	0.05	0.07	0.06	135
haa	0.28	0.16	0.20	135
jeem	0.19	0.24	0.21	135
kaaf	0.19	0.12	0.15	135
khaa	0.28	0.30	0.29	135
la	0.12	0.12	0.12	135
laam	0.07	0.05	0.06	132
meem	0.10	0.10	0.10	135
nun	0.14	0.14	0.14	123
ra	0.11	0.08	0.09	123
saad	0.19	0.21	0.20	135
seen	0.24	0.36	0.29	135
sheen	0.37	0.37	0.37	135
ta	0.13	0.06	0.08	135
taa	0.08	0.02	0.04	135
thaa	0.18	0.13	0.15	137
thal	0.10	0.13	0.11	135
toot	0.09	0.09	0.09	135
waw	0.08	0.21	0.12	121
ya	0.20	0.24	0.22	134
yaa	0.14	0.19	0.16	135
zay	0.09	0.07	0.08	135
accuracy			0.16	4247
macro avg	0.16	0.16	0.15	4247
weighted avg	0.16	0.16	0.16	4247

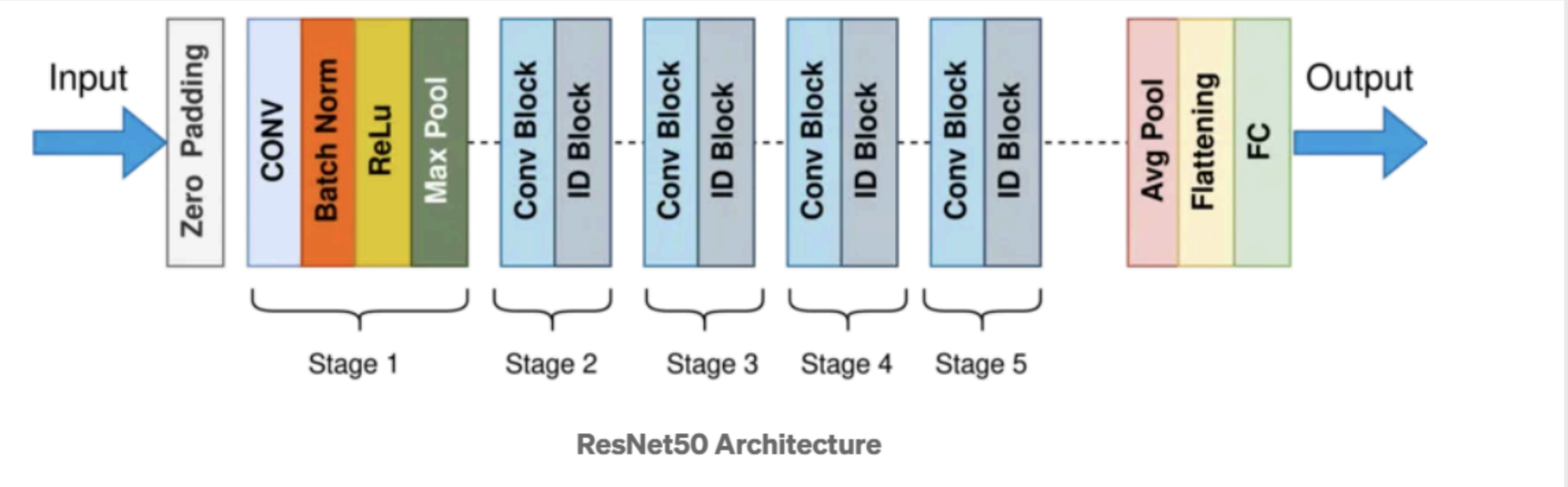


Model 2: Rsnet50



- ResNet50 is a deep convolutional neural network introduced by Microsoft Research in 2015.
- The name “50” indicates that the architecture contains 50 layers.
- It uses residual connections to bypass layers and ease optimization.
- These residual pathways help the model avoid vanishing gradients, enabling much deeper networks than earlier CNN architectures.
- The residual design allows the network to learn refined residual functions rather than full mappings.

Architecture of Rsnet50

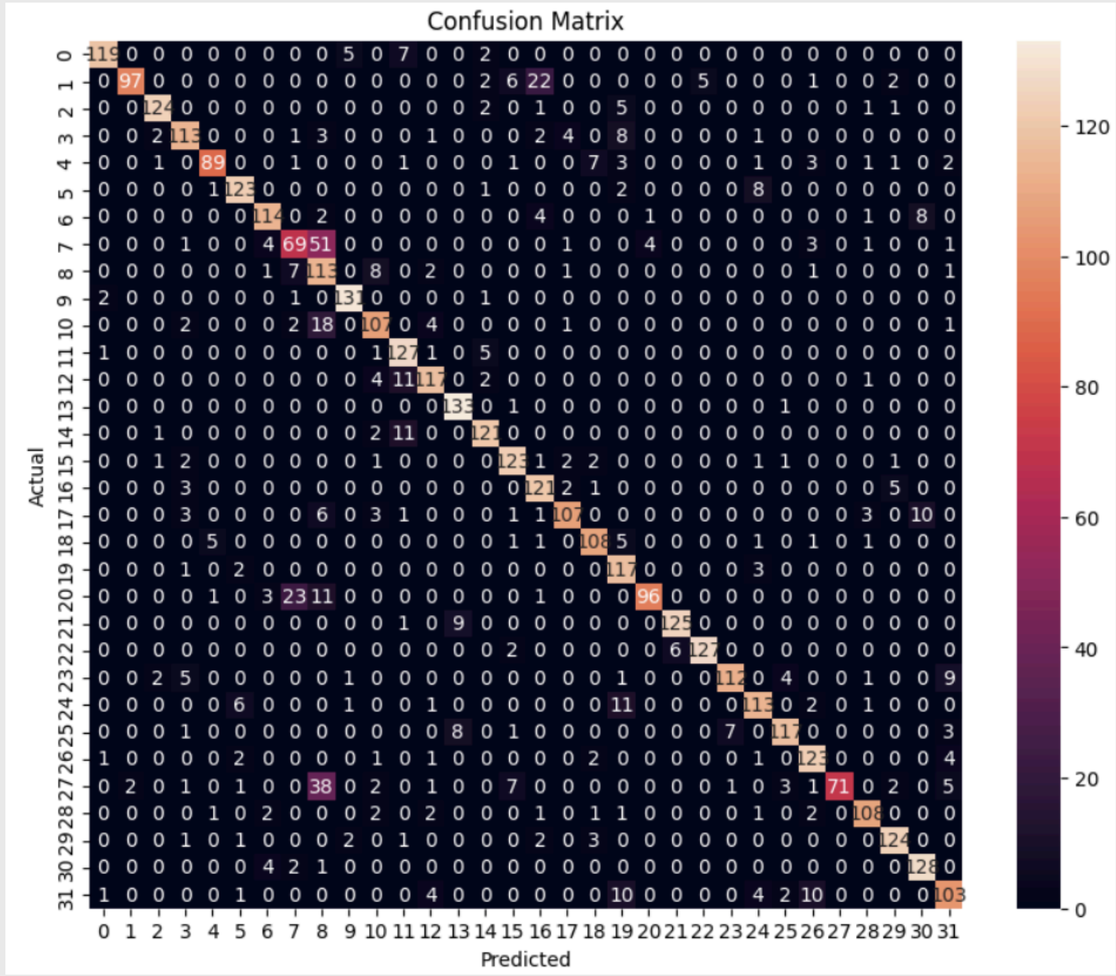


Result of the Rsnet50:

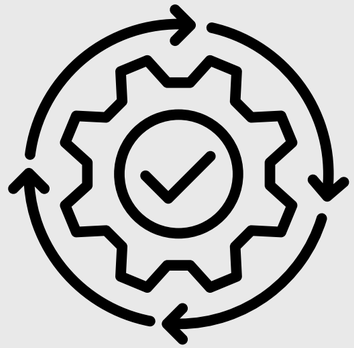
```
Training Accuracy:
Accuracy: 92.90%
Validation Accuracy:
Accuracy: 85.24%
```

Classification Report:

	precision	recall	f1-score	support
ain	0.96	0.89	0.93	133
al	0.98	0.72	0.83	135
aleff	0.95	0.93	0.94	134
bb	0.85	0.84	0.84	135
dal	0.92	0.80	0.86	111
dha	0.90	0.91	0.91	135
dhad	0.89	0.88	0.88	130
fa	0.65	0.51	0.57	135
gaaf	0.47	0.84	0.60	134
ghain	0.94	0.97	0.95	135
ha	0.82	0.79	0.80	135
haa	0.79	0.94	0.86	135
jeem	0.87	0.87	0.87	135
kaaf	0.89	0.99	0.93	135
khaa	0.89	0.90	0.89	135
la	0.86	0.91	0.88	135
laam	0.77	0.92	0.84	132
meem	0.91	0.79	0.85	135
nun	0.87	0.88	0.87	123
ra	0.72	0.95	0.82	123
saad	0.95	0.71	0.81	135
seen	0.95	0.93	0.94	135
sheen	0.96	0.94	0.95	135
ta	0.93	0.83	0.88	135
taa	0.84	0.84	0.84	135
thaa	0.91	0.85	0.88	137
thal	0.84	0.91	0.87	135
toot	1.00	0.53	0.69	135
waw	0.91	0.89	0.90	121
ya	0.91	0.93	0.92	134
yaa	0.88	0.95	0.91	135
zay	0.80	0.76	0.78	135
accuracy			0.85	4247
macro avg	0.87	0.85	0.85	4247
weighted avg	0.87	0.85	0.85	4247

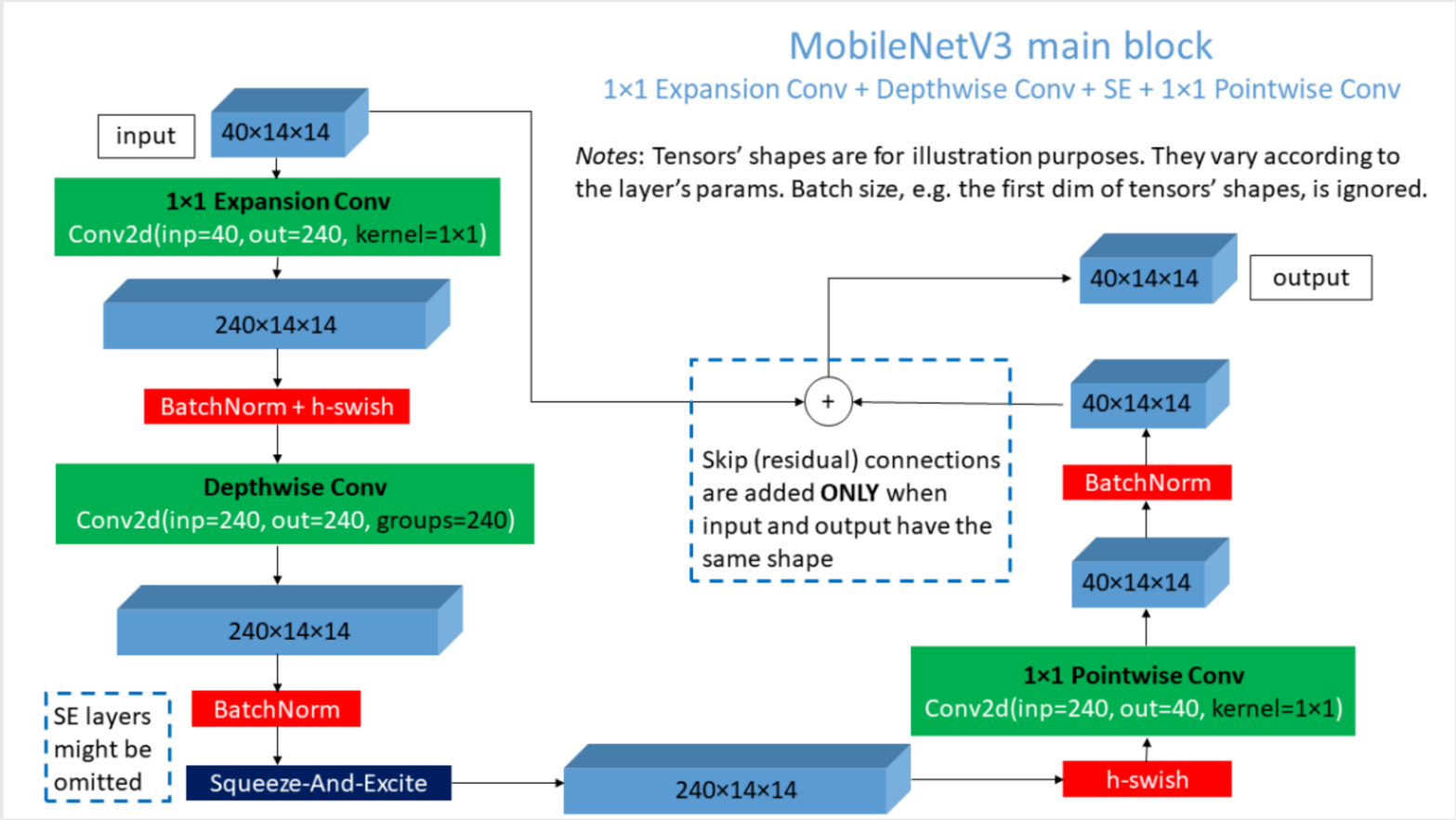


Model 3: MobileNetV3



- A lightweight convolutional neural network optimized for mobile and edge devices
- Designed using Neural Architecture Search (NAS) and NetAdapt for hardware-aware efficiency
- Uses depthwise separable convolutions to reduce computation
- Integrates Squeeze-and-Excitation (SE) blocks for better feature attention
- Introduces efficient activations (Hard-Swish, Hard-Sigmoid) to cut latency

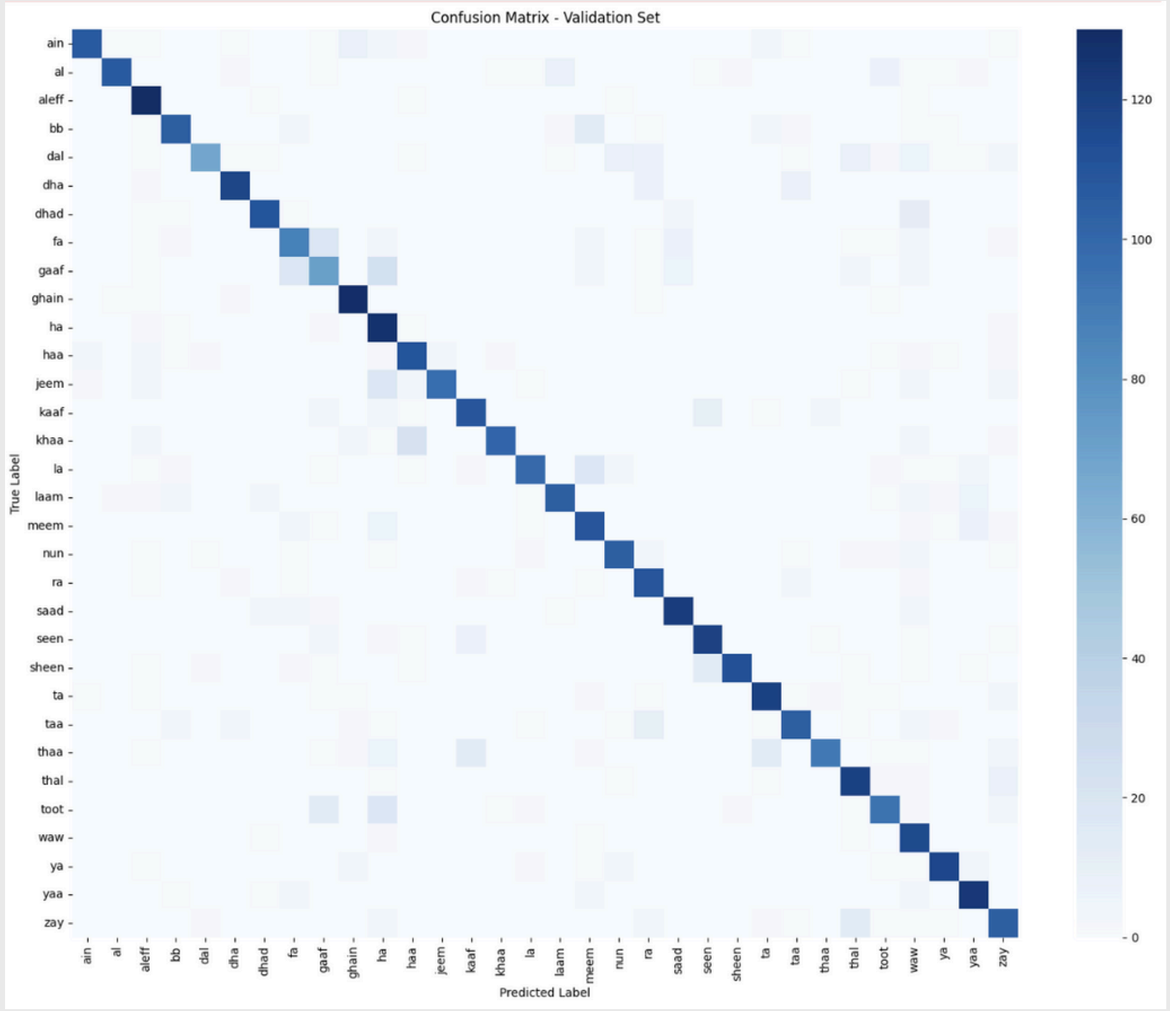
Architecture of MobileNetV3



Result of the MobileNetV3:

Classification Report:

	precision	recall	f1-score	support
ain	0.94	0.81	0.87	133
al	0.96	0.79	0.87	135
aleff	0.81	0.97	0.88	134
bb	0.88	0.79	0.83	135
dal	0.91	0.60	0.72	111
dha	0.91	0.87	0.89	135
dhad	0.91	0.85	0.88	130
fa	0.69	0.65	0.67	135
gaaf	0.58	0.53	0.55	134
ghain	0.87	0.96	0.91	135
ha	0.57	0.94	0.71	135
haa	0.76	0.82	0.79	135
jeem	0.96	0.73	0.83	135
kaaf	0.82	0.81	0.81	135
khaa	0.95	0.75	0.84	135
la	0.90	0.73	0.81	135
laam	0.89	0.80	0.84	132
meem	0.70	0.81	0.75	135
nun	0.85	0.85	0.85	123
ra	0.74	0.89	0.81	123
saad	0.88	0.90	0.89	135
seen	0.83	0.88	0.85	135
sheen	0.97	0.83	0.89	135
ta	0.81	0.89	0.85	135
taa	0.85	0.78	0.81	135
thaa	0.94	0.66	0.78	137
thal	0.77	0.89	0.82	135
toot	0.80	0.70	0.75	135
waw	0.62	0.96	0.76	121
ya	0.91	0.87	0.89	134
yaa	0.82	0.92	0.87	135
zay	0.73	0.78	0.75	135
accuracy			0.81	4247
macro avg	0.83	0.81	0.81	4247
weighted avg	0.83	0.81	0.81	4247



Deployment of our model

Repository:
https://github.com/AlreemA/Arabic-Sign-Language

Demo Link (Streamlit Cloud Deployment):
https://arabic-sign-language-csbp411-group8.streamlit.app/

Future Improvements in Customizing The CNN

- Add more advanced augmentations (blur, cutout, mixup) for better generalization.
- Attempting to turn the images black and white so the model focus more on the hand
- Using the box labels in the dataset to show the model where it should focus on in training
- Implement caching and prefetching to speed up training

Resources Used:

- https://medium.com/@nitishkundu1993/exploring-resnet50-an-in-depth-look-at-the-model-architecture-and-code-implementation-d8d8fa67e46f
- https://medium.com/@RobuRishabh/understanding-and-implementing-mobilenetv3-422bd0bdfb5a
- https://medium.com/@myringoleMLGOD/simple-convolutional-neural-network-cnn-for-dummies-in-pytorch-a-step-by-step-guide-6f4109f6df80
- https://github.com/JayPatwardhan/ResNet-PyTorch/blob/master/ResNet/CIFAR10_ResNet50.ipynb
- Dataset:
- https://www.kaggle.com/datasets/ammarsayedtaha/arabic-sign-language-dataset-2022/data

Q/A

THANK YOU