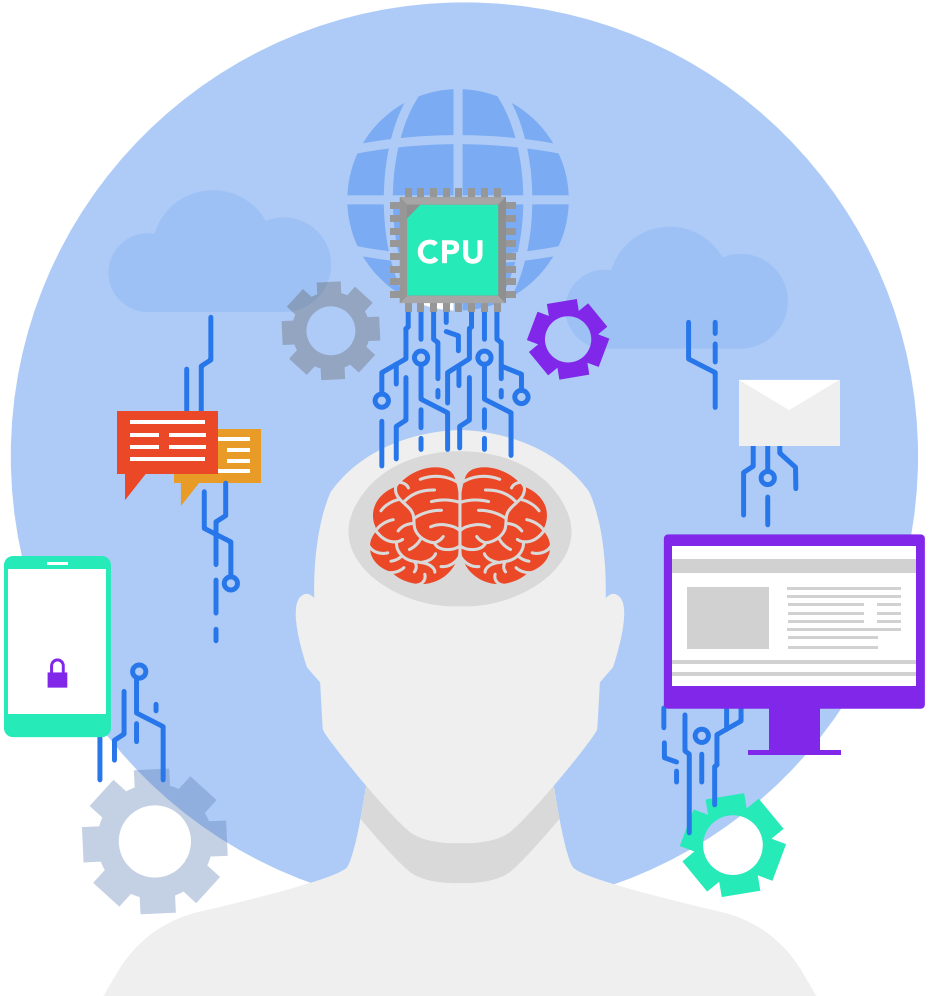


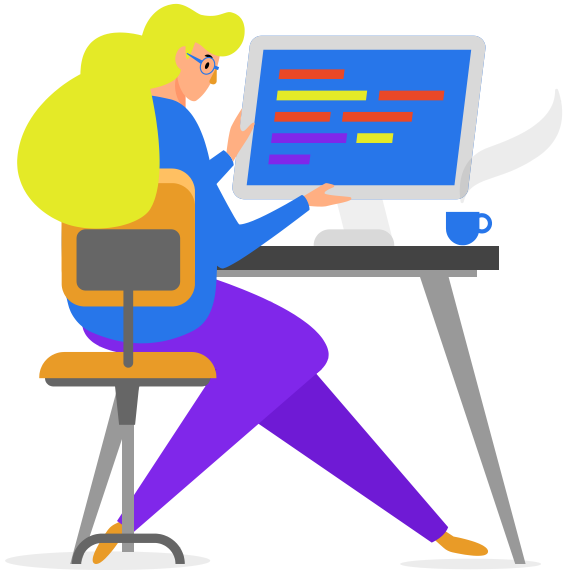
Université Saint-Joseph de Beyrouth
Faculté des sciences

Machine Learning & Deep Learing

Prof: Dr Youssef Bakouny

Done by:
Ahmad Hussein

Machine Learning Iterations



01

Iteration 1

Traditional Algorithms & Deep Learning on numeric Dataset

02

Iteartion 2

Image Classification, Auto encoders & Clustering

03

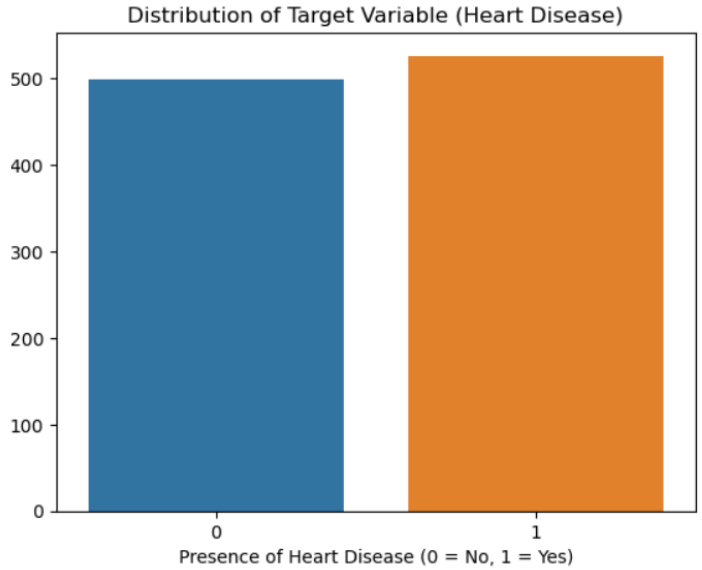
Iteration 3 &4

Final Project:
Sign Language Translation

Iteration 1 – Traditional Algorithms

The Heart Disease Dataset

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0




```
# Converting categorical variables to dummy variables
cp = pd.get_dummies(data['cp'], prefix='cp', drop_first=True)
restecg = pd.get_dummies(data['restecg'], prefix='restecg', drop_first=True)
slope = pd.get_dummies(data['slope'], prefix='slope', drop_first=True)
ca = pd.get_dummies(data['ca'], prefix='ca', drop_first=True)
thal = pd.get_dummies(data['thal'], prefix='thal', drop_first=True)
```

Iteration 1 – Traditional Algorithms

The Heart Disease Dataset

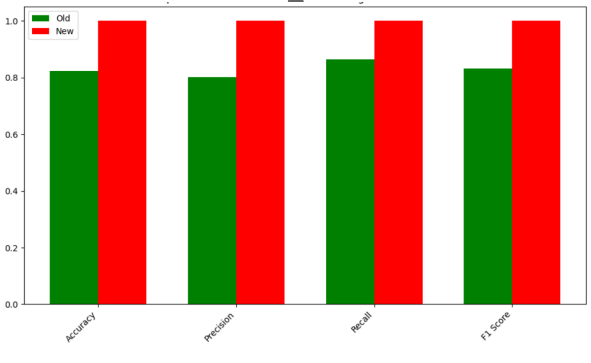
Model	Accuracy	Precision	Recall	F1 score
Logistic Regression	0.824390	0.781513	0.902913	0.837838
Naïve Bayes	0.814634	0.787611	0.864078	0.824074
Random Forest	0.990244	0.980952	1.000000	0.990385
Extreme Gradient Boost	0.946341	0.950980	0.941748	0.946341
K-Nearest Neighbor	0.824390	0.801802	0.864078	0.831776
Decision Tree	0.980488	0.962617	1.000000	0.980952
Support Vector Machine	0.863415	0.826087	0.922330	0.871560

Iteration 1 – HyperParameter Tuning

K- Nearest Neighbor

01

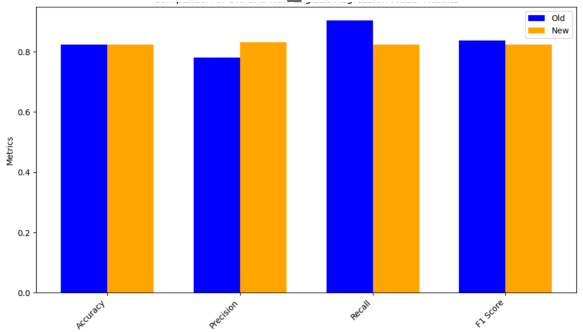
```
knn_params = {
    'n_neighbors': [3, 5, 7, 9],
    'weights': ['uniform', 'distance']
}
```



Logistic Regression

02

```
log_reg_params = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'penalty': ['l2']
}
```



Iteration 1 – Deep Learning Models

Functional API Model

- Accuracy: 84.87%
- Precision: 81.03%
- Recall: 91.26%

Vs

Sequential Model

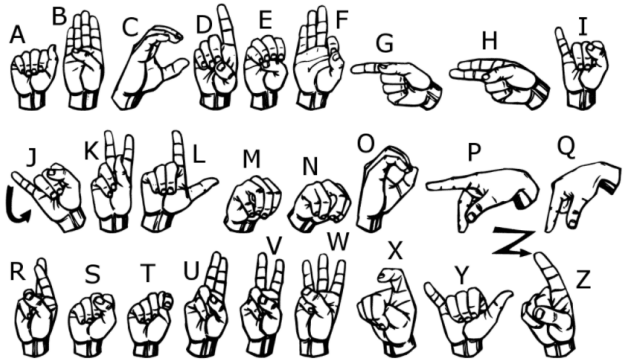
- Accuracy: 90.73%
- Precision: 87.5%
- Recall: 95.15%

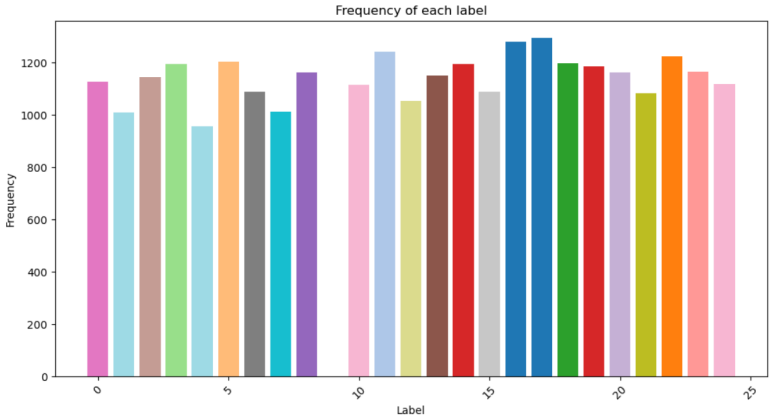
Iteration 2 – Image Classification

MNIST Sign Language Dataset

SVM Model
- 86% accuracy

CNN
- 96% accuracy





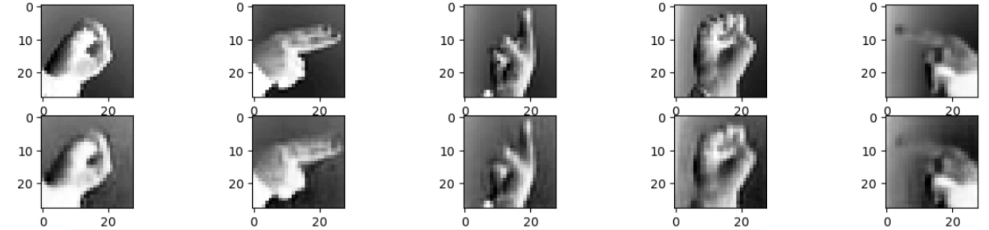
Iteration 2 – Image Classification

Iteration 2 - Autoencoders

Encoder

01 Reduce the dimensionality of the input data while preserving important features

```
reconstruct_img(autoencoder2, X_test, 5)
1/1 [==============================] - 0s 22ms/step
```

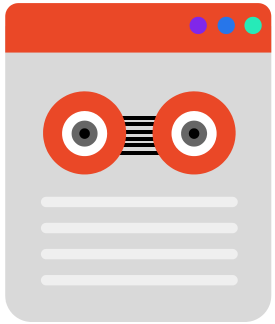


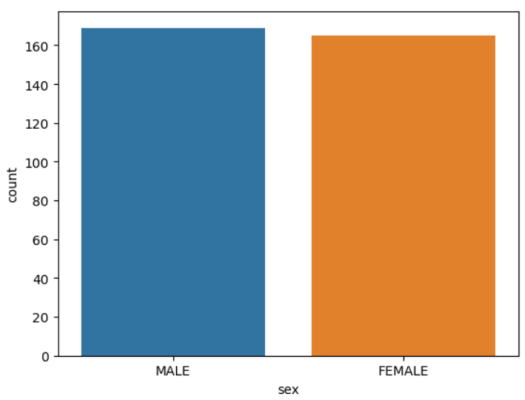
Decoder

02 Reconstruct the input data from the encoded representation produced by the encoder

Iteration 2 - Clustering

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	39.1	18.7	181.0	3750.0	MALE
1	39.5	17.4	186.0	3800.0	FEMALE
2	40.3	18.0	195.0	3250.0	FEMALE
3	NaN	NaN	NaN	NaN	NaN
4	36.7	19.3	193.0	3450.0	FEMALE





Data Cleaning & Prep

Removing Nan values and preparing data for model

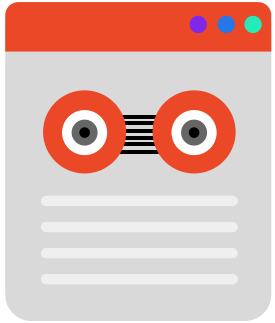
```
scaler = StandardScaler()
X = scaler.fit_transform(df)
df_scaled = pd.DataFrame(data=X, columns=df.columns)
```

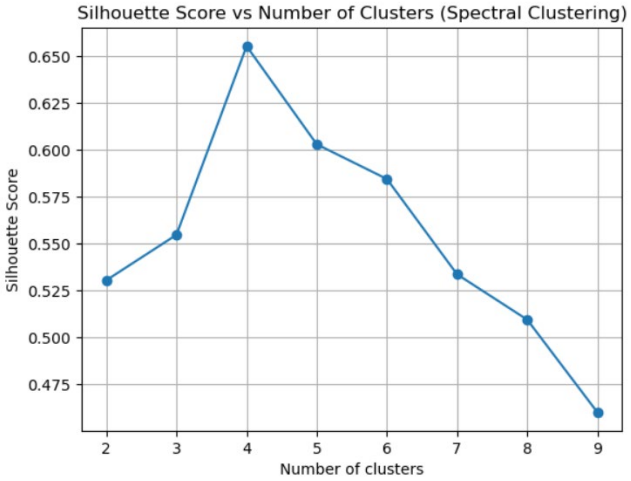
```
pca = PCA(n_components=2)

pca.fit(df_scaled)
df_pca = pca.transform(df_scaled)
```

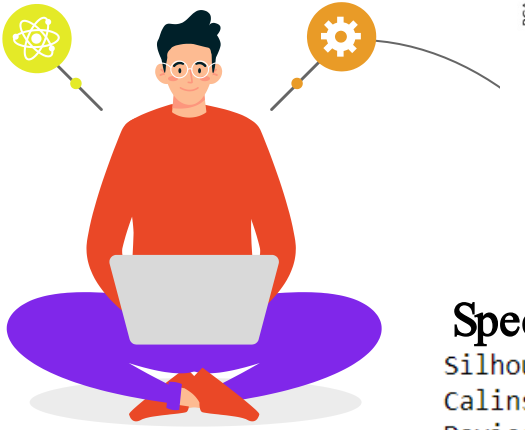
Iteration 2 - Clustering

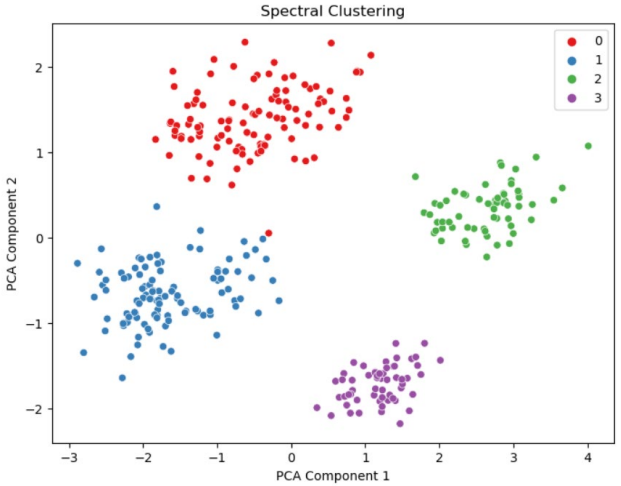
The model

```
silhouette_scores = []

for k in range(2, 10):
    spectral = SpectralClustering(n_clusters=k, random_state=42)
    labels = spectral.fit_predict(df_pca)

    silhouette_scores.append(silhouette_score(df_pca, labels))
```





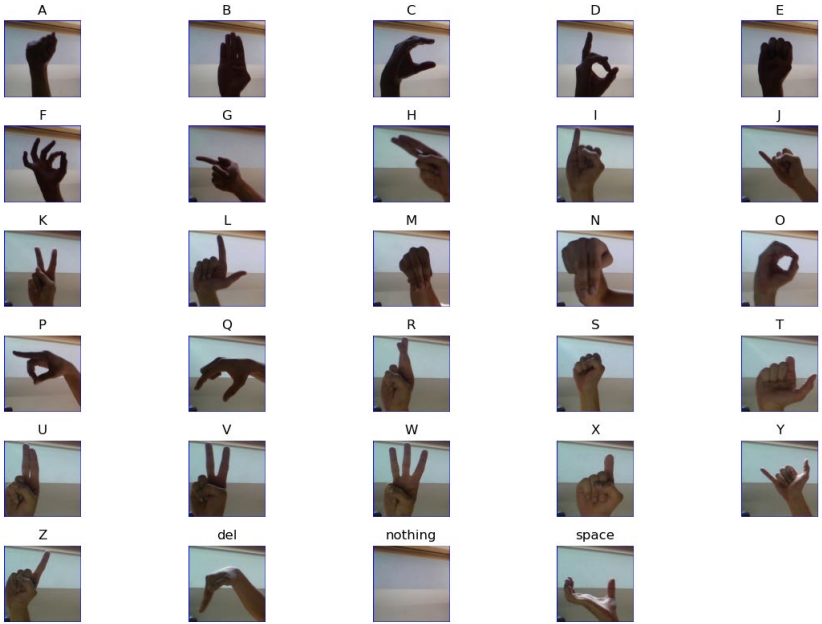




Spectral Clustering

Silhouette Score: 0.6554730690324767
Calinski-Harabasz Index: 916.403631169708
Davies-Bouldin Index: 0.45495707952834175

Iteration 3 – ASL Translation




```python
def main():
    while True:
        audio = listen_and_recognize()
        text = recognize_speech(audio)

        if text:
            print("Speaker:", text)
            if text.lower() == 'quit':
                break
            else:
                # Saving the recognized text
                return text

if __name__ == "__main__":
    recognized_text = main()
    print("Recognized text:", recognized_text)
```

```
Speak now
Sorry, I couldn't understand what you said.
Speak now
Speaker: hello
Recognized text: hello
```

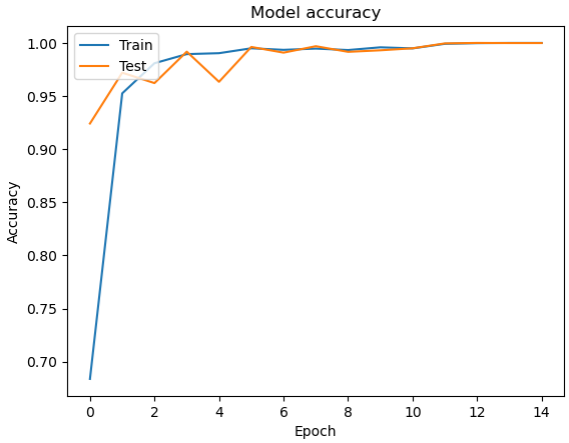
Iteration 3 – ASL Translation

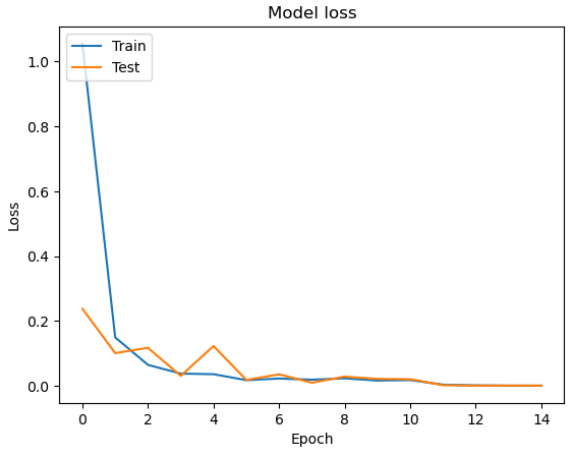
```python
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),

    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),

    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(29, activation='softmax')
])
```

```python
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

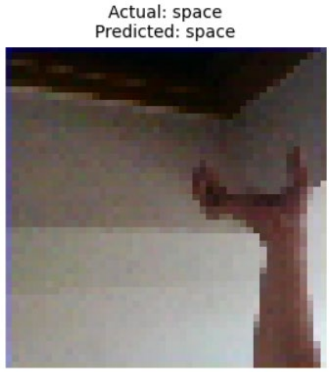
```python
history = model.fit(train_generator,
                    epochs=15,
                    validation_data=validation_generator)
```

```
Test Accuracy: 99.99%
```






```
1/1 [==============================] - 0s 21ms/step
```



Iteration 3 – ASL Translation

```python
# Initializing Mediapipe Hands
mphands = mp.solutions.hands
hands = mphands.Hands()
mp_drawing = mp.solutions.drawing_utils

# Opening
cap = cv2.VideoCapture(0)

# Checking webcam opened
if not cap.isOpened():
    print("Error: Failed to open v
    exit()

_, frame = cap.read()
h, w, c = frame.shape

img_counter = 0
```

```python
while True:
    # Reading frame from webcam
    ret, frame = cap.read()

    # Checking frame is read
    if not ret:
        print("Error: Failed to capture frame.")
        break

    k = cv2.waitKey(1)
    if k%256 == 27:
        # ESC to close
        print("Escape hit, closing...")
        break
    elif k%256 == 32:
        # SPACE to predict and print predictions
        analysisframe = frame
        showframe = analysisframe
        cv2.imshow("Frame", showframe)

        framergbanalysis = cv2.cvtColor(analysisframe, cv2.COLOR_BGR2RGB)
        resultanalysis = hands.process(framergbanalysis)
        hand_landmarksanalysis = resultanalysis.multi_hand_landmarks
        if hand_landmarksanalysis:
```

```python
if hand_landmarksanalysis:
    for handLMsanalysis in hand_landmarksanalysis:
        #dimensions of the bounding box to be arounf the hand
        x_max = 0
        y_max = 0
        x_min = w
        y_min = h
        for lmanalysis in handLMsanalysis.landmark:
            x, y = int(lmanalysis.x * w), int(lmanalysis.y * h)
            if x > x_max:
                x_max = x
            if x < x_min:
                x_min = x
            if y > y_max:
                y_max = y
            if y < y_min:
                y_min = y
        # Adding some padding to the bounding box
        y_min -= 20
        y_max += 20
        x_min -= 20
        x_max += 20
```

Iteration 3 – ASL Translation

```
Analysis Frame Shape: (48, 48, 3)
1/1 [==============================] - 0s 24ms/step
Predicted Character :  A
Confidence :  71.98681235313416
Analysis Frame Shape: (48, 48, 3)
1/1 [==============================] - 0s 22ms/step
Predicted Character :  D
Confidence :  32.79445171356201
Escape hit, closing...
```

```
pred
```

```
'AD'
```

```
import pyttsx3

text_to_speech = pyttsx3.init()

voices = text_to_speech.getProperty('voices')
text_to_speech.setProperty('voice', voices[1].id) # 0 for male and 1 for female

# convert text to speech
text_to_speech.say(pred)

# save the audio file
text_to_speech.save_to_file(pred, 'test.mp3')

# listen to audio
text_to_speech.runAndWait()
```

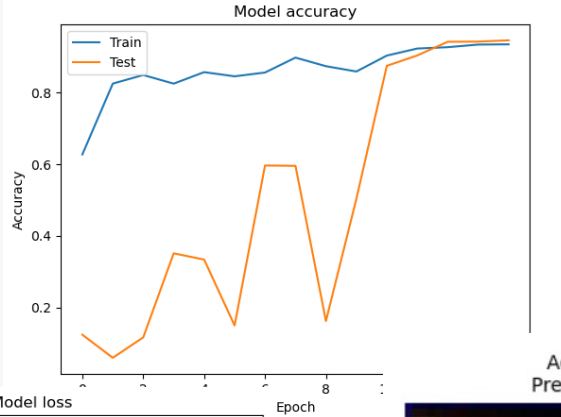
Iteration 4 – English to Arabic sign translation

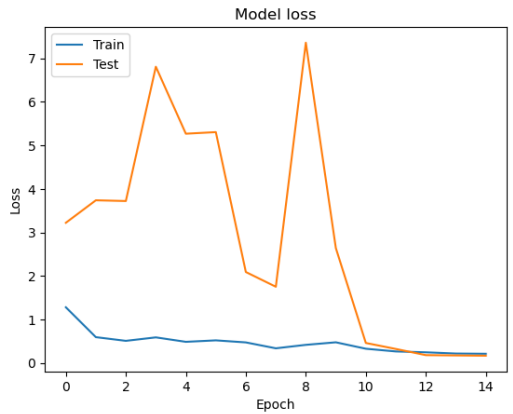
```python
from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.layers import Dropout, Dense, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model

# Loading pre-trained MobileNetV2 model
base_model = MobileNetV2(input_shape=(pic_size, pic_size, 3), include_top=False, weights='imagenet')

# Unfreezing some layers in the base model
for layer in base_model.layers[-20:]:
    layer.trainable = True

# Custom classification layers
x = Flatten()(base_model.output)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
output = Dense(len(labels), activation='softmax')(x)

# The fine-tuned model
model = Model(inputs=base_model.input, outputs=output)
```

```python
# Learning rate scheduler
def lr_schedule(epoch):
    initial_lr = 0.001
    if epoch < 10:
        return initial_lr
    else:
        return initial_lr * 0.1  # to reduce learning rate by *0.1 after 10 epochs

lr_scheduler = LearningRateScheduler(lr_schedule)

model.compile(optimizer=Adam(learning_rate=0.0001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```python
history = model.fit(train_generator,
                    epochs=15,
                    validation_data=validation_generator,
                    callbacks=[lr_scheduler])
```

```
Test Accuracy: 97.38%
```





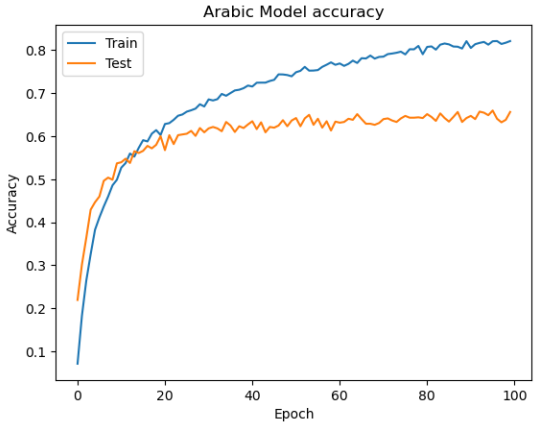
Actual: Y
Predicted: Y

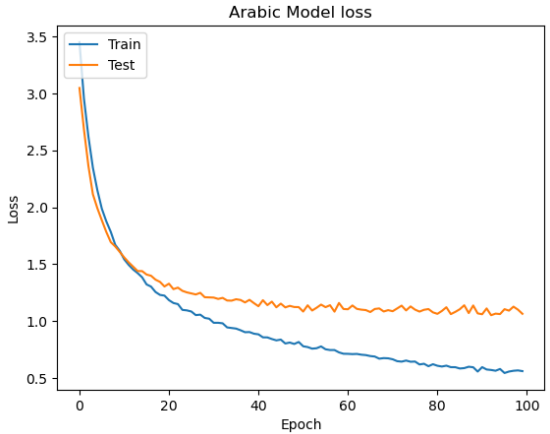
Iteration 4 – English to Arabic sign translation

```python
import tensorflow as tf
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model

num_classes = 31
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freezing the base model layers
base_model.trainable = False

# custom classification layers
x = GlobalAveragePooling2D()(base_model.output)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(num_classes, activation='softmax')(x)

arabic_model = Model(inputs=base_model.input, outputs=x)
```

```python
arabic_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
Test Accuracy: 71.63%
```





Iteration 4 – English to Arabic sign translation

	english	arabic
0	Hi.	.مرحبًا
1	Run!	!اركض
2	Help!	!النجدة
3	Jump!	!اقفز
4	Stop!	!قف
...	...	...
24633	rising voices promoting a more linguistically ...	...شاركنا تحدي ابداع ميم بلغتك الام تعزيزا للتنوع
24634	following last year s successful campaign we i...	...استكمالا لنجاح حملة العام السابق ندعوكم للمشار
24635	during last year s challenge we also met langu...	...تعرفنا خلال تحدي العام الماضي على ابطال لغويين
24636	to take part just follow the simple steps outl...	...للمشاركة في التحدي اتبع الخطوات الموضحة على ال
24637	you will also find links to some free web base...	...ستجد ايضا روابط لمجموعة من منصات ابداع الميم ا

```
# english tokenizer
english_tokenizer=Tokenizer()
english_tokenizer.fit_on_texts(data["english"])

vocab_size_english=len(english_tokenizer.word_index)
english_word_2_idx=english_tokenizer.word_index
english_idx_2_word={idx:word for word,idx in english_word_2_idx.items()}
```

```
print(english_idx_2_word)
```

```
{1: 'the', 2: 'of', 3: 'to
14: 'by', 15: 'global', 16
e', 25: 'he', 26: 'we', 27
6: 'my', 37: 'do', 38: 'w
```

```
# arabic tokenizer
arabic_tokenizer=Tokenizer()
arabic_tokenizer.fit_on_texts(data["arabic"])
vocab_size_arabic=len(arabic_tokenizer.word_index)+1

arabic_word_2_idx=arabic_tokenizer.word_index
arabic_idx_2_word={idx:word for word,idx in arabic_word_2_idx.items()}
```

```
vocab_size_arabic
```

```
54770
```

```
print(arabic_word_2_idx)
```

```
': 6, 'الاصوات': 7, 'الى': 8, 'لا': 9, 'هذا': 10, 'ما': 11, 'التي': 12, 'هذه': 13,
نت': 19, 'او': 20, 'كان': 21, 'يوم': 22, 'و': 23, 'هل': 24, 'كل': 25, 'بعد': 2
'عام': 32, 'العالم': 33, 'حيث': 34, 'كما': 35, 'هناك': 36, 'المدون': 37, 'هنا': 38,
```

Iteration 4 – English to Arabic sign translation

```
# Pad sequences transforming words to theor respective integers numbers(tokens)

token_eng=english_tokenizer.texts_to_sequences(data["english"])
token_ara=arabic_tokenizer.texts_to_sequences(data["arabic"])

padded_eng=pad_sequences(token_eng,maxlen=50,padding="post")
padded_ara=pad_sequences(token_ara,maxlen=50,padding="post")
```

```
padded_eng.shape, padded_ara.shape
```

```
((24407, 50), (24407, 50))
```

```
padded_eng[35]
```

```
array([139,   8,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,
```

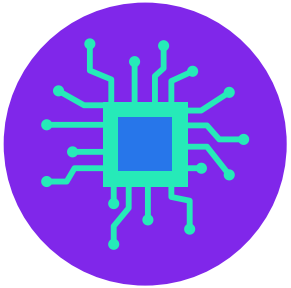
```
tanslation_model=Sequential()

tanslation_model.add(Embedding(vocab_size_english, 50, input_length=50))
tanslation_model.add(tf.keras.layers.Bidirectional(LSTM(units=128)))
tanslation_model.add(tf.keras.layers.RepeatVector(50))
tanslation_model.add(LSTM(128,return_sequences=True))
tanslation_model.add(TimeDistributed(Dense(vocab_size_arabic,activation="softmax")))
```

```
# Compile the model
tanslation_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
# Train the model with reduced batch size
history = tanslation_model.fit(X_train, y_train, batch_size=128, validation_split=0.2, verbose=2, epochs=50)
```

```
from transformers import pipeline

pipe = pipeline("translation_en_to_ar", model="marefa-nlp/marefa-mt-en-ar")
```

Thank you!



Resources:

- Heart disease dataset: https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset
- Mnist sign dataset: https://www.kaggle.com/datasets/datamunge/sign-language-mnist?resource=download
- Penguin dataset: https://www.kaggle.com/datasets/youssefaboelwafa/clustering-penguins-species
- ASL alphabet dataset: https://www.kaggle.com/datasets/grassknoted/asl-alphabet/data
- Arabic sign dataset: https://www.kaggle.com/datasets/muhammadalbrham/rgb-arabic-alphabets-sign-language-dataset/data
- Translation dataset: https://www.kaggle.com/datasets/samirmoustafa/arabic-to-english-translation-sentences
- Power point template : Slidesgo