AI/ML

AI PROJECT

EXPLORE TRANSFER LEARNING THROUGH SMALL AND LARGE MODELS

BY: IKRAM ACHALI / LOKESHWARN VENGADABADY

CONTENT

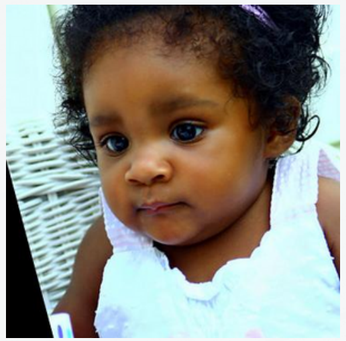
TRANSFER LEARNING

- Pre-Trained Knowledge Reuse
- Fine-Tuning
- Efficiency and Performance

DATA COLLECTION

7,591 images (of age 1-100) with associated real and apparent age labels

APPA-REAL DATABASE



Ex 1 :
age : 1

Ex 2 :
age : 60

Model Used:
NASNet & Mobilenet v3 Small

Why did we do this choice ?

1. State-of-the-Art Accuracy: these models deliver high performance on complex datasets like ImageNet.
2. Efficient Transfer Learning: Pre-trained on ImageNet, it provides robust feature extraction.
3. Scalability: Adjustable architecture for balancing accuracy and computational cost.

NASNET
TRANSFER

LET’S DISCOVER THE PROCESS

Preprocessing

IMAGES

PREPROCESSING

7,591 images with associated real and apparent age labels

APPA-REAL DATABASE

Read images in the dataset and save them in array format in the npy files

IMAGES TO NPY FILES

Image arrays to be between 0 and 255

READY TO BE USED

```
### Train CSV ###
    file_name  num_ratings  apparent_age_avg  apparent_age_std  real_age
0  000000.jpg           36          5.000000          1.146423         4
1  000001.jpg           63         20.079365          4.096819        18
2  000002.jpg           38         76.815789          6.133009        80
3  000003.jpg           38         55.657895          7.864653        50
4  000004.jpg           15         17.666667          3.457222        17


### Test CSV ###
    file_name  num_ratings  apparent_age_avg  apparent_age_std  real_age
0  005613.jpg           39         23.205128          5.530678        19
1  005614.jpg           38         70.736842          6.570549        76
2  005615.jpg           38         55.368421          5.874457        40
3  005616.jpg           36         24.277778          2.224681        21
4  005617.jpg           39         25.230769          5.691460        34


### Valid CSV ###
    file_name  num_ratings  apparent_age_avg  apparent_age_std  real_age
0  004113.jpg           26         26.230769          4.003076        29
1  004114.jpg           39         27.256410          4.552142        25
2  004115.jpg           14         23.142857          7.262730        37
3  004116.jpg           38         73.289474          6.472010        80
4  004117.jpg           21         20.142857          2.797958        25
```

```python
if (generate_X == True):
    for i in train_img_names[:Nb_train]:
        img = Image.open(f'{base_dir}/train/{i}').convert("RGB")
        img = img.resize((224,224))
        img_array = np.array(img)
        image_list.append(img_array)
        if (float(i.strip('.jpg')) % 1000==0):
            print(f"completed {i.strip('.jpg')} images")
    np.save(f'{base_dir}/data_train.npy', image_list)
```

```python
base_model = tf.keras.applications.NASNetMobile(input_shape=(224, 224, 3), weights='imagenet', include_top=False)
preprocess_input_Nasnet = tf.keras.applications.nasnet.preprocess_input
Nasnet_model = tf.keras.models.Model(inputs=base_model.input, outputs=output)
X_train_processed_Nasnet = preprocess_input_Nasnet(np.copy(X_train))
X_valid_processed_Nasnet = preprocess_input_Nasnet(np.copy(X_valid))


base_model1 = tf.keras.applications.MobileNetV3Small(input_shape=(224, 224, 3), weights='imagenet', include_top=Fals
preprocess_input_Mobilenet = tf.keras.applications.mobilenet_v3.preprocess_input
X_train_processed_Mobilenet = preprocess_input_Mobilenet(np.copy(X_train))
X_valid_processed_Mobilenet = preprocess_input_Mobilenet(np.copy(X_valid))
```

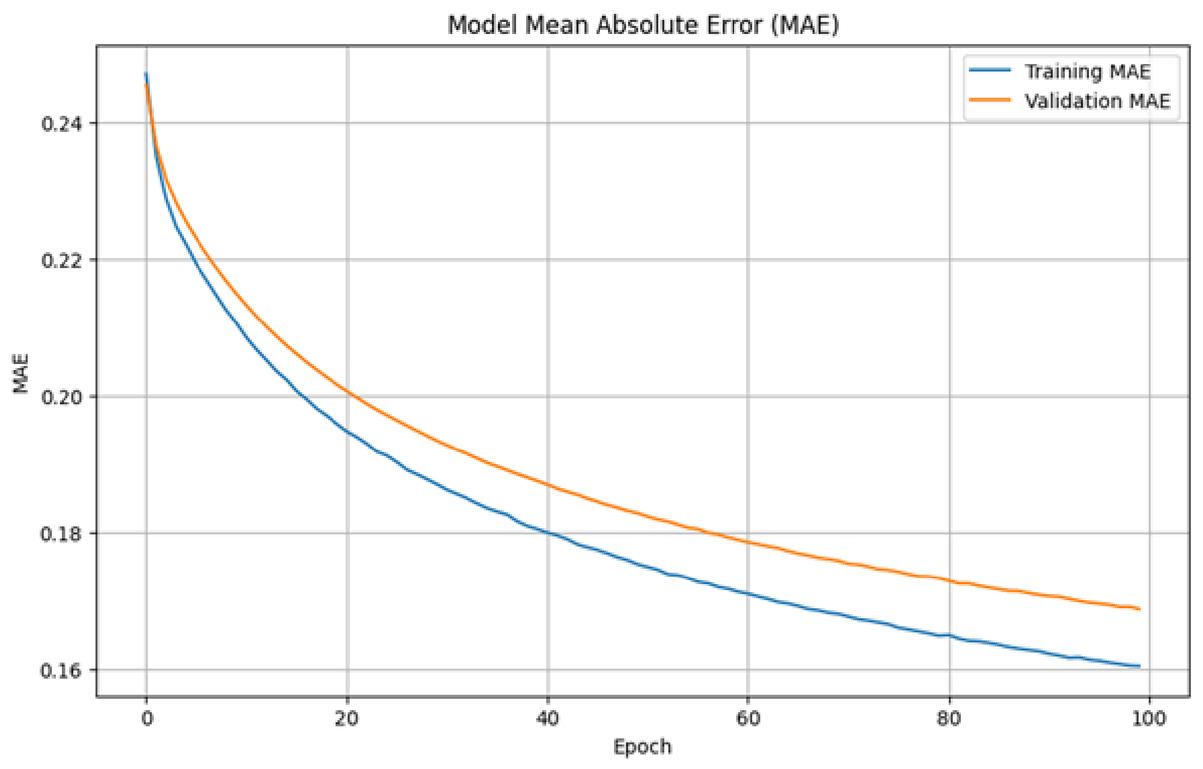
Train Model

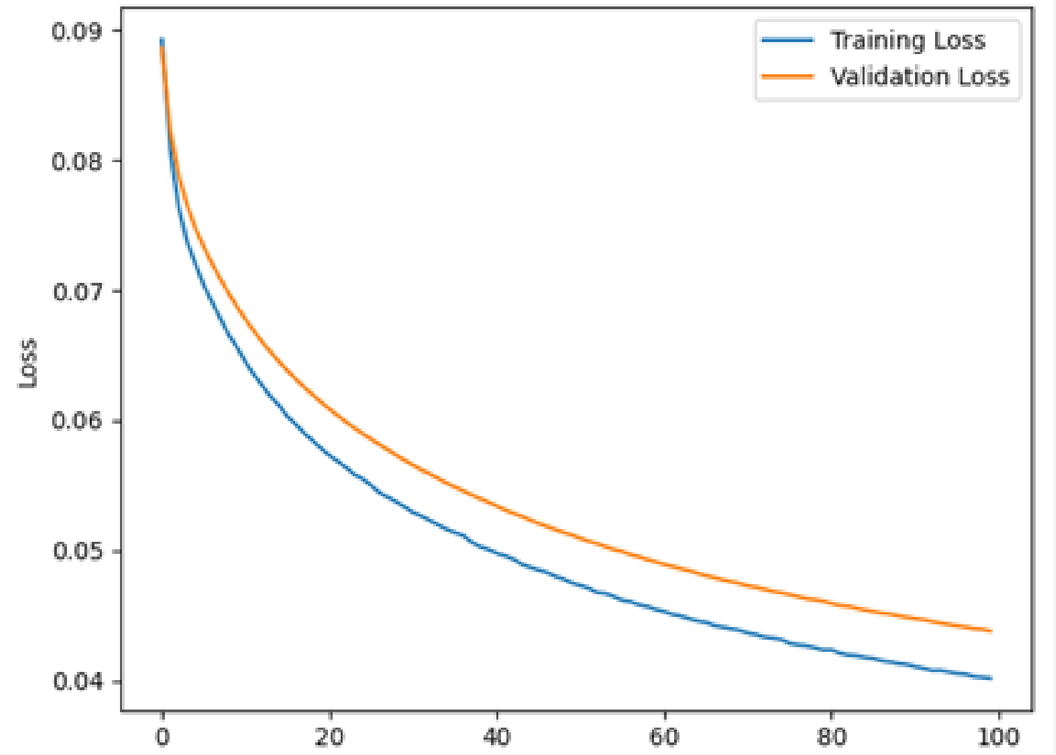
ORIGINAL MODEL + ONE OUTPUT LAYER

Nasnet

```
[12] ### Nasnet basic
     early_stopping_Nasnet = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=6)
     # Load NASNetMobile with pre-trained ImageNet weights, excluding the top layer
     base_model = tf.keras.applications.NASNetMobile(input_shape=(224, 224, 3), weights='imagenet', include_top=False)
     #preprocess_input = tf.keras.applications.mobilenet.preprocess_input
     preprocess_input_Nasnet = tf.keras.applications.nasnet.preprocess_input
     base_model.trainable = False
     # Adding custom layers
     x = base_model.output
     output = tf.keras.layers.Dense(1, activation='sigmoid')(x)
     # Create the new model with NASNetMobile as the base
     Nasnet_model = tf.keras.models.Model(inputs=base_model.input, outputs=output)
     # Preprocess the data
     X_train_processed_Nasnet = preprocess_input_Nasnet(np.copy(X_train))
     X_valid_processed_Nasnet = preprocess_input_Nasnet(np.copy(X_valid))
     print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))

     Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/nasnet/NASNet-mobile-no-top.h5
     19993432/19993432 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
     Num GPUs Available:  1
```

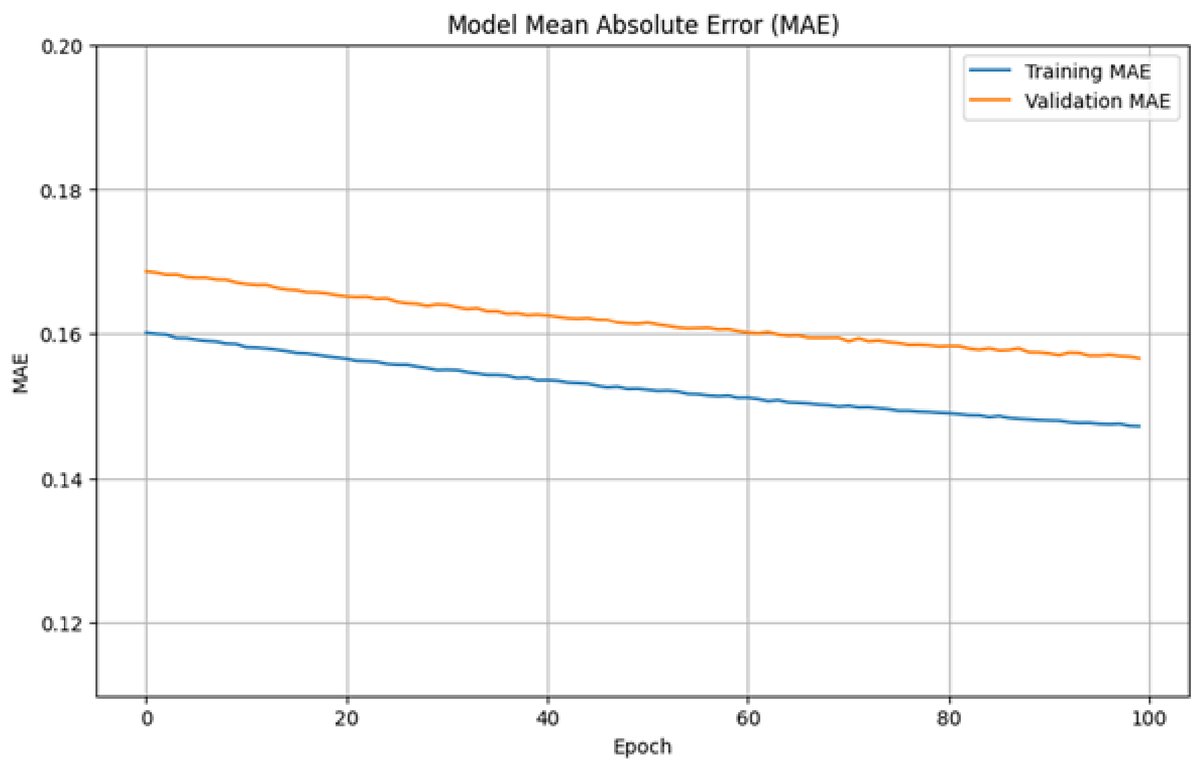
Nasnet

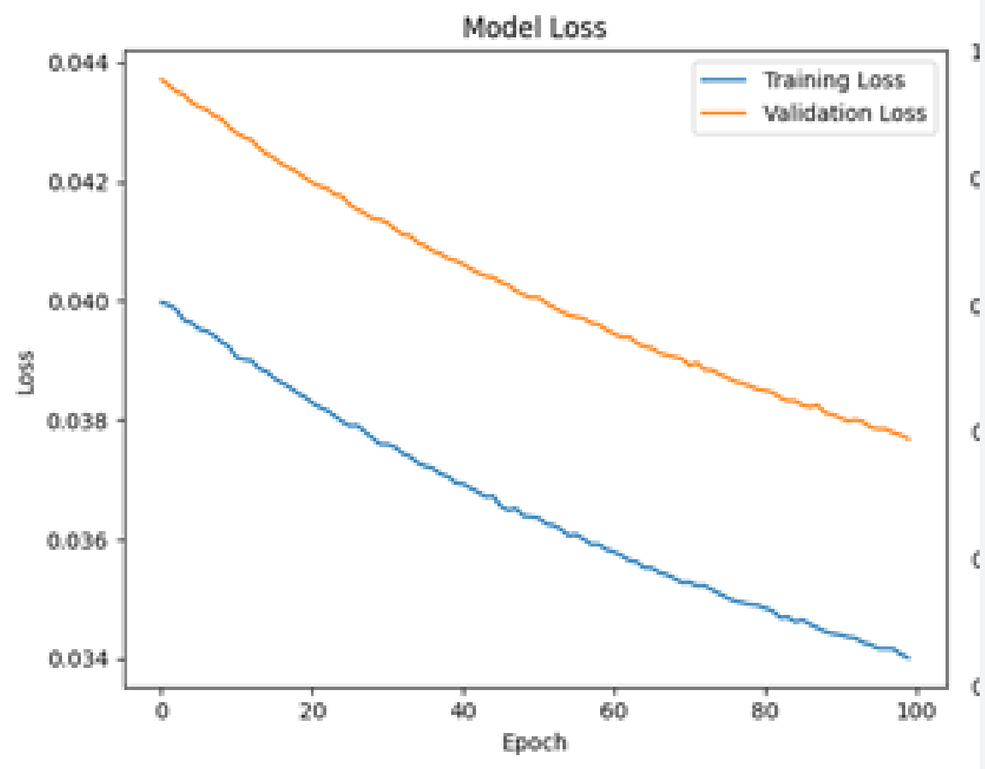

Model Mean Absolute Error (MAE)
Training MAE
Validation MAE
MAE
Epoch
0.24
0.22
0.20
0.18
0.16
0
20
40
60
80
100


Training Loss
Validation Loss
Loss
0.09
0.08
0.07
0.06
0.05
0.04
0
20
40
60
80
100

Nasnet

WITH MORE EPOCH

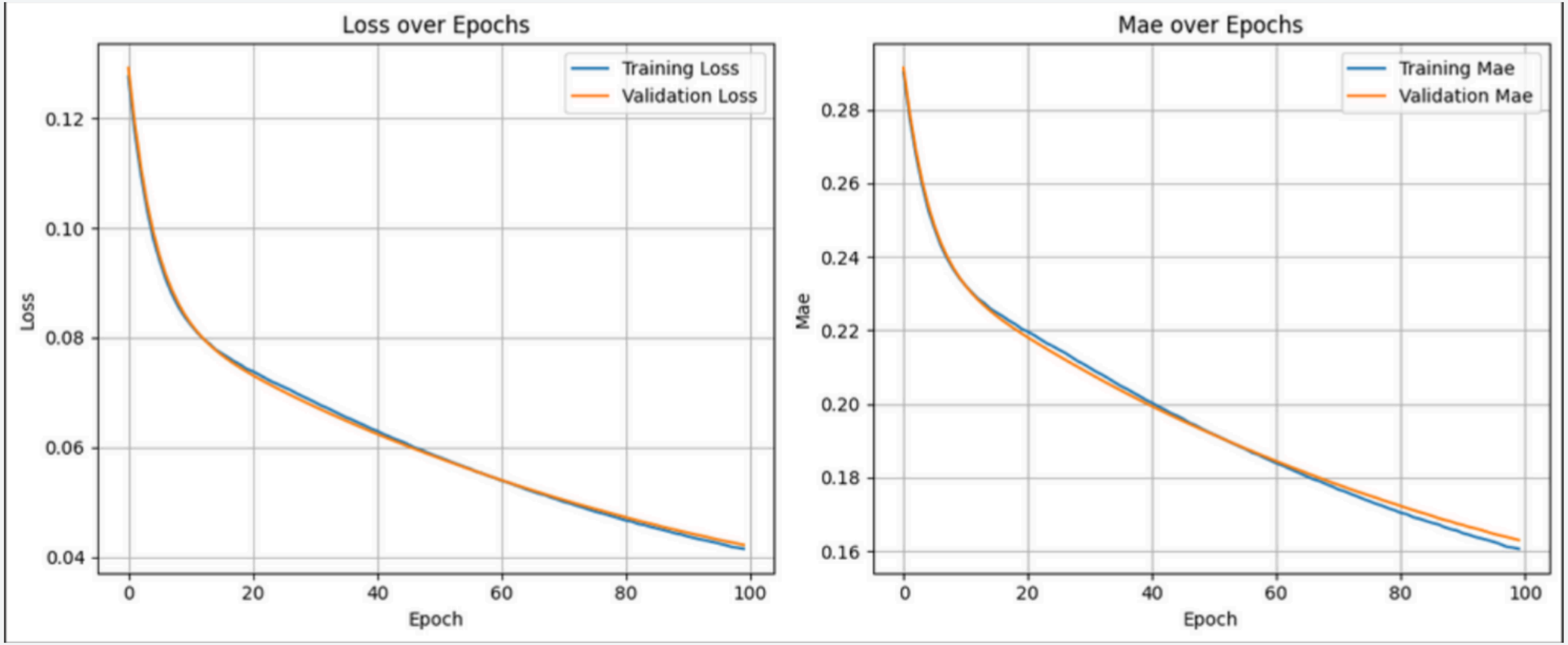




Mobilenet v3 Small

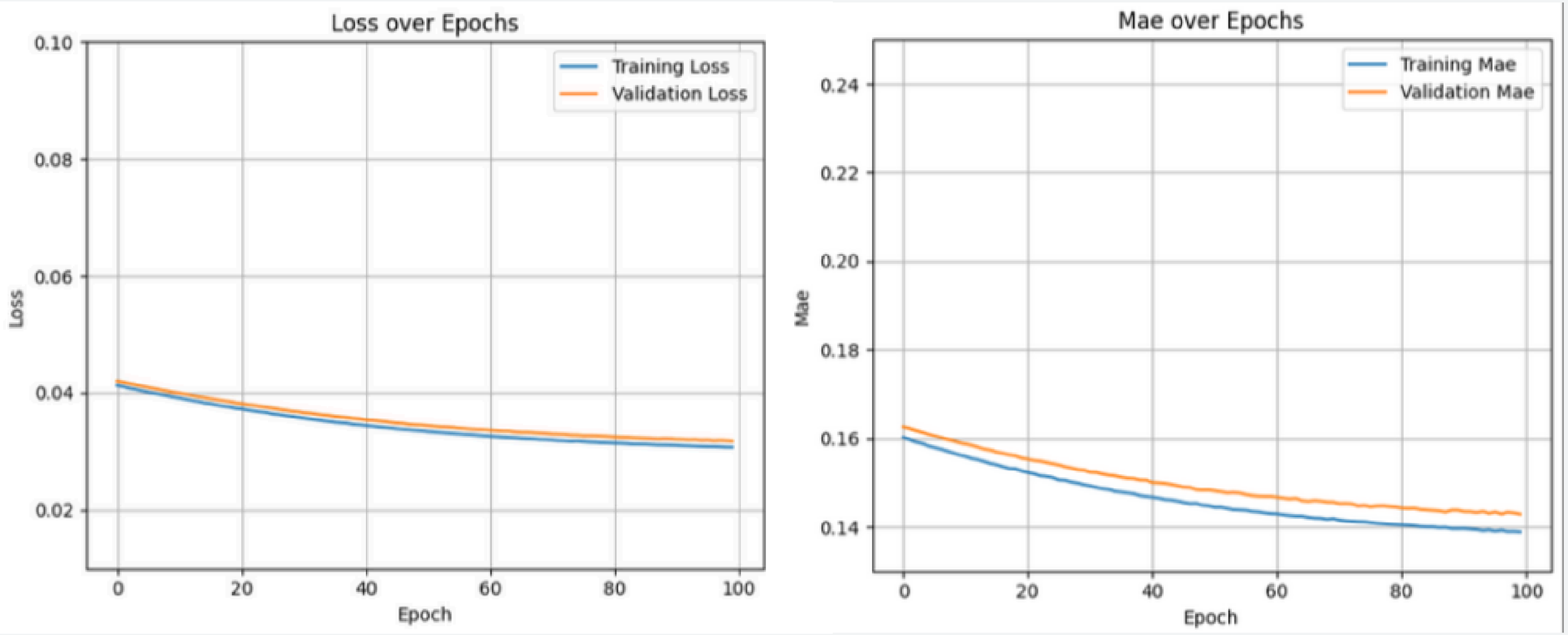
```
[19] ### Mobilenet basic
     early_stopping_MobileNet = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=7)

     #base_model = tf.keras.applications.ResNet50(input_shape=(224, 224, 3), weights='imagenet', include_top=False)
     base_model = tf.keras.applications.MobileNetV3Small(input_shape=(224, 224, 3), weights='imagenet', include_top=False)

     #preprocess_input = tf.keras.applications.mobilenet.preprocess_input
     preprocess_input_Mobilenet = tf.keras.applications.mobilenet_v3.preprocess_input

     base_model.trainable = False

     # Add custom layers
     #x = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
     #x = tf.keras.layers.Dense(1024, activation='relu')(x)
     #x = tf.keras.layers.Dropout(0.5)(x)
     output = tf.keras.layers.Dense(1, activation='sigmoid')(base_model.output)

     # Create the new model with MobileNet as the base
     Mobilenet_model = tf.keras.models.Model(inputs=base_model.input, outputs=output)
     # Preprocess the data
     X_train_processed_Mobilenet = preprocess_input_Mobilenet(np.copy(X_train))
     X_valid_processed_Mobilenet = preprocess_input_Mobilenet(np.copy(X_valid))
```

Mobilenet v3 Small



Mobilenet v3 Small

WITH MORE EPOCH



Train Model

UNFREEZED LAYERS+ GLOBAL AVERAGE POOLING & RELU LAYERS

```python
### Nasnet implementation avec 0.7 de Dropout and with 1 trainable layer
early_stopping_Nasnet = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=6)
# Load NASNetMobile with pre-trained ImageNet weights, excluding the top layer
base_model = tf.keras.applications.NASNetMobile(input_shape=(224, 224, 3), weights='imagenet', include_top=False)
#preprocess_input = tf.keras.applications.mobilenet.preprocess_input
preprocess_input_Nasnet = tf.keras.applications.nasnet.preprocess_input
base_model.trainable = False
for layer in base_model.layers[-1:]:
    layer.trainable = True
# Adding custom layers
x = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
x = tf.keras.layers.Dense(1024, activation='relu')(x)
x = tf.keras.layers.Dropout(0.7)(x)  # Dropout with 50% probability
output = tf.keras.layers.Dense(1, activation='sigmoid')(x)
# Create the new model with NASNetMobile as the base
Nasnet_model = tf.keras.models.Model(inputs=base_model.input, outputs=output)
# Preprocess the data
X_train_processed_Nasnet = preprocess_input_Nasnet(np.copy(X_train))
X_valid_processed_Nasnet = preprocess_input_Nasnet(np.copy(X_valid))
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

```
Epoch 47/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 4s 34ms/step - loss: 0.0156 - mae: 0.0982 - val_loss: 0.0217 - val_mae: 0.1155
Epoch 48/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 4s 34ms/step - loss: 0.0164 - mae: 0.0999 - val_loss: 0.0216 - val_mae: 0.1160
Epoch 49/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 4s 34ms/step - loss: 0.0150 - mae: 0.0960 - val_loss: 0.0216 - val_mae: 0.1154
Epoch 50/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 4s 34ms/step - loss: 0.0143 - mae: 0.0942 - val_loss: 0.0216 - val_mae: 0.1152
Epoch 51/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 4s 35ms/step - loss: 0.0138 - mae: 0.0926 - val_loss: 0.0216 - val_mae: 0.1151
Epoch 52/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 4s 35ms/step - loss: 0.0147 - mae: 0.0944 - val_loss: 0.0216 - val_mae: 0.1153
Epoch 53/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 4s 35ms/step - loss: 0.0139 - mae: 0.0919 - val_loss: 0.0216 - val_mae: 0.1155
Epoch 54/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 4s 35ms/step - loss: 0.0140 - mae: 0.0933 - val_loss: 0.0216 - val_mae: 0.1157
Epoch 55/100
```

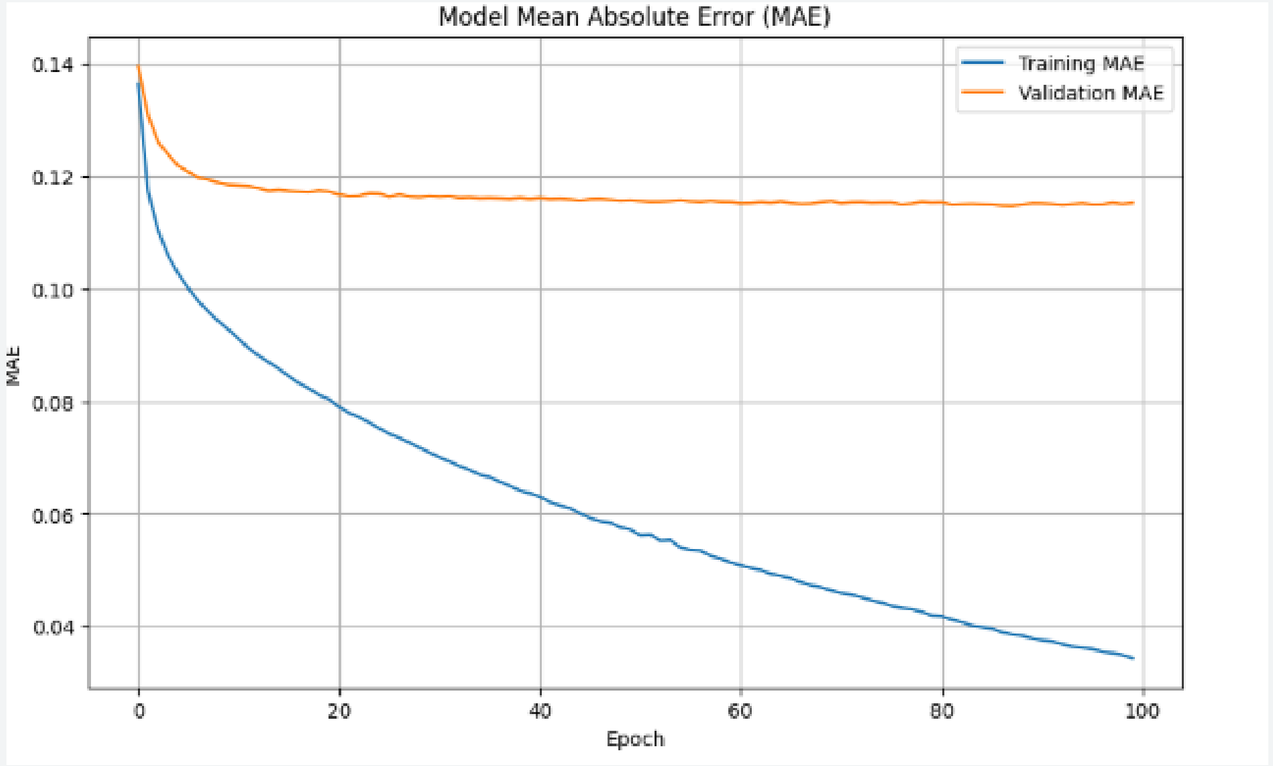
```
[12] ### Mobilenet basic
     early_stopping_MobileNet = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=7)
     #base_model = tf.keras.applications.ResNet50(input_shape=(224, 224, 3), weights='imagenet', include_top=False)
     base_model = tf.keras.applications.MobileNetV3Small(input_shape=(224, 224, 3), weights='imagenet', include_top=False)
     #preprocess_input = tf.keras.applications.mobilenet.preprocess_input
     preprocess_input_Mobilenet = tf.keras.applications.mobilenet_v3.preprocess_input
     base_model.trainable = False
     base_model.trainable = False
     for layer in base_model.layers[-1:]:
         layer.trainable = True

     # Add custom layers
     x = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
     x = tf.keras.layers.Dense(1024, activation='relu')(x)
     x = tf.keras.layers.Dropout(0.7)(x)
     output = tf.keras.layers.Dense(1, activation='sigmoid')(x)

     # Create the new model with MobileNet as the base
     Mobilenet_model = tf.keras.models.Model(inputs=base_model.input, outputs=output)
     # Preprocess the data
     X_train_processed_Mobilenet = preprocess_input_Mobilenet(np.copy(X_train))
     X_valid_processed_Mobilenet = preprocess_input_Mobilenet(np.copy(X_valid))
```

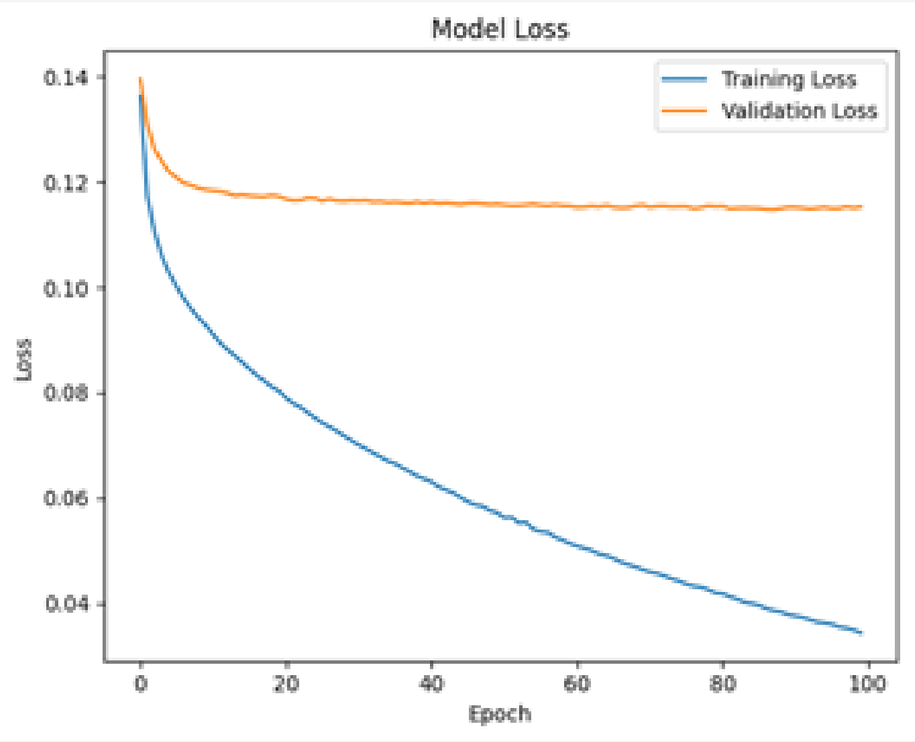
```
125/125 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0124 - mae: 0.0876 - val_loss: 0.0176 - val_mae: 0.1028
Epoch 95/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0129 - mae: 0.0894 - val_loss: 0.0176 - val_mae: 0.1030
Epoch 96/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0127 - mae: 0.0898 - val_loss: 0.0176 - val_mae: 0.1028
Epoch 97/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0126 - mae: 0.0883 - val_loss: 0.0176 - val_mae: 0.1027
Epoch 98/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0126 - mae: 0.0888 - val_loss: 0.0176 - val_mae: 0.1032
Epoch 99/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0123 - mae: 0.0880 - val_loss: 0.0176 - val_mae: 0.1031
Epoch 100/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0126 - mae: 0.0878 - val_loss: 0.0176 - val_mae: 0.1029
```

Train Model

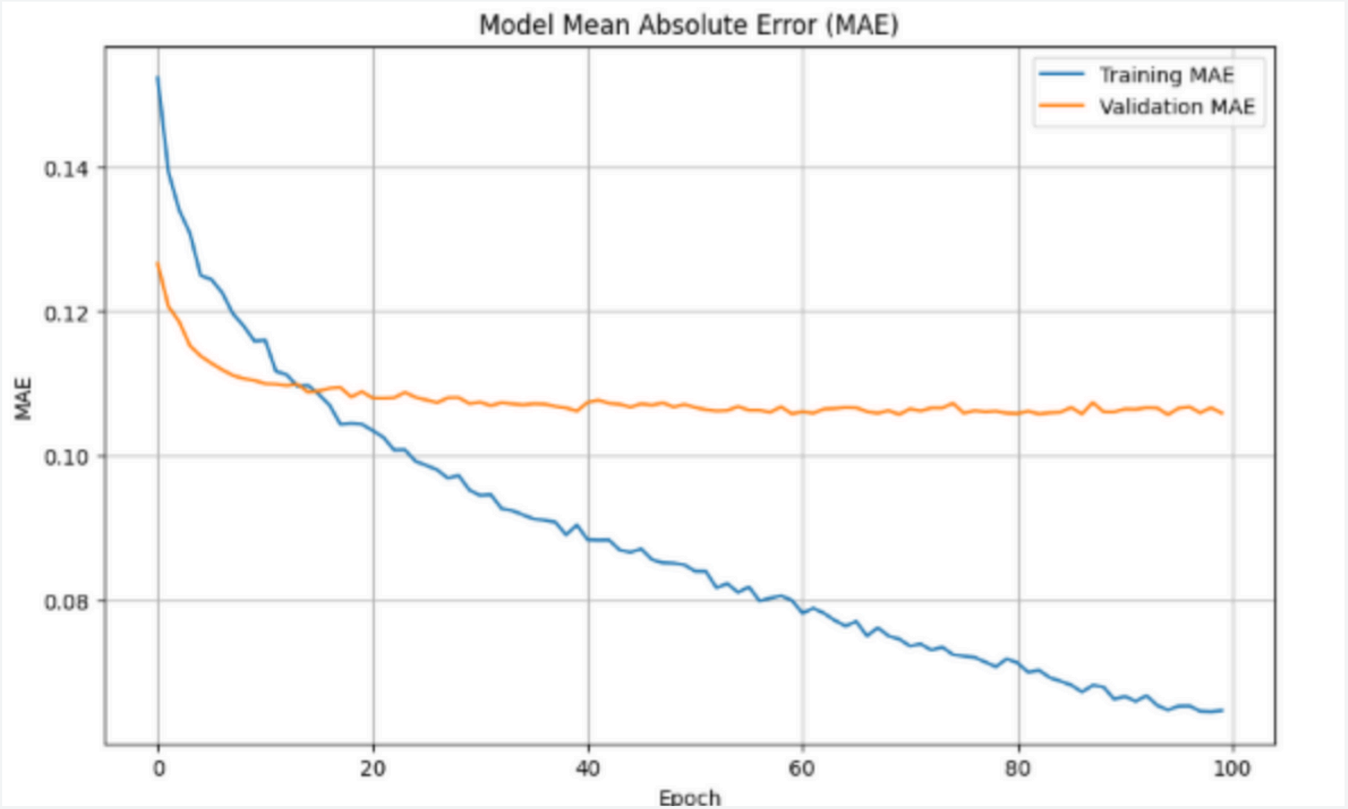
UNFREEZE MORE

```
early_stopping_MobileNet = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=5)

#base_model = tf.keras.applications.ResNet50(input_shape=(224, 224, 3), weights='imagenet', include_top=False)
base_model = tf.keras.applications.MobileNetV3Small(input_shape=(224, 224, 3), weights='imagenet', include_top=False)

#preprocess_input = tf.keras.applications.mobilenet.preprocess_input
preprocess_input_Mobilenet = tf.keras.applications.mobilenet_v3.preprocess_input

base_model.trainable = False
for layer in base_model.layers[-3:]:
    layer.trainable = True

# Add custom layers
x = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
x = tf.keras.layers.Dense(1024, activation='relu')(x)
output = tf.keras.layers.Dense(1, activation='sigmoid')(x)

# Create the new model with NASNetMobile as the base
Mobilenet_model = tf.keras.models.Model(inputs=base_model.input, outputs=output)
# Preprocess the data
X_train_processed_Mobilenet = preprocess_input_Mobilenet(np.copy(X_train))
X_valid_processed_Mobilenet = preprocess_input_Mobilenet(np.copy(X_valid))

print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```


Model Mean Absolute Error (MAE)
Training MAE
Validation MAE
MAE
Epoch
0.14
0.12
0.10
0.08
0.06
0.04
0
20
40
60
80
100


Model Loss
Training Loss
Validation Loss
Loss
Epoch
0.14
0.12
0.10
0.08
0.06
0.04
0
20
40
60
80
100

```
early_stopping_MobileNet = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=3)

#base_model = tf.keras.applications.ResNet50(input_shape=(224, 224, 3), weights='imagenet', include_top=False)
base_model = tf.keras.applications.MobileNetV3Small(input_shape=(224, 224, 3), weights='imagenet', include_top=False)

#preprocess_input = tf.keras.applications.mobilenet.preprocess_input
preprocess_input_Mobilenet = tf.keras.applications.mobilenet_v3.preprocess_input

base_model.trainable = False
for layer in base_model.layers[-3:]:
    layer.trainable = True

# Add custom layers
x = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
x = tf.keras.layers.Dense(1024, activation='relu')(x)
x = tf.keras.layers.Dropout(0.5)(x)
output = tf.keras.layers.Dense(1, activation='sigmoid')(x)

# Create the new model with NASNetMobile as the base
Mobilenet_model = tf.keras.models.Model(inputs=base_model.input, outputs=output)
# Preprocess the data
X_train_processed_Mobilenet = preprocess_input_Mobilenet(np.copy(X_train))
X_valid_processed_Mobilenet = preprocess_input_Mobilenet(np.copy(X_valid))

print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```


Model Mean Absolute Error (MAE)
Training MAE
Validation MAE
0.14
0.12
0.10
0.08
MAE
0
20
40
60
80
100
Epoch

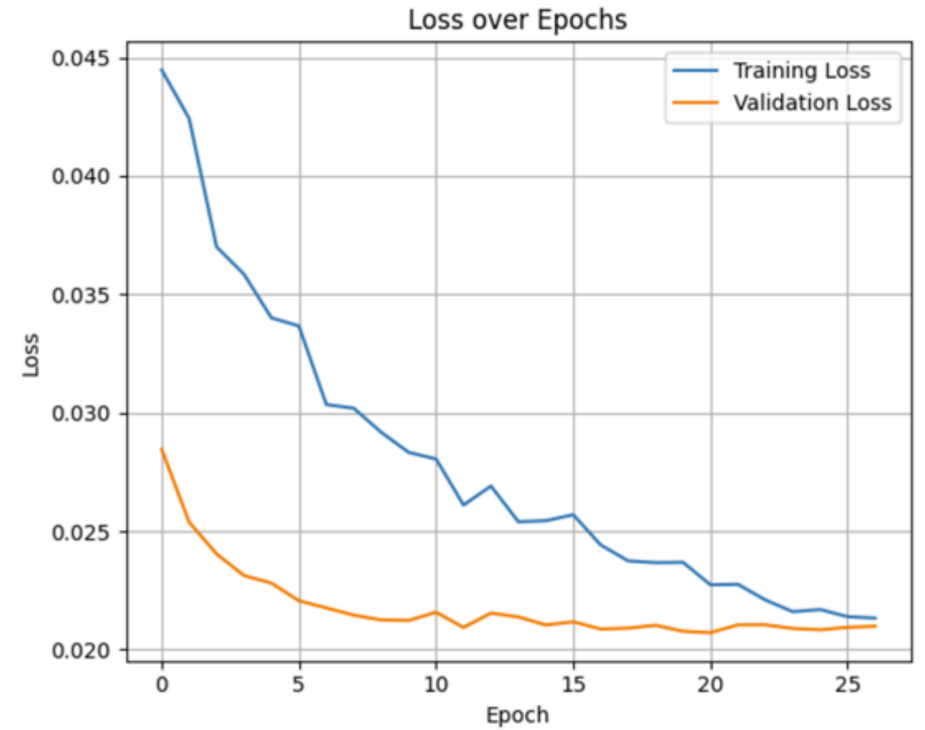
Train Model

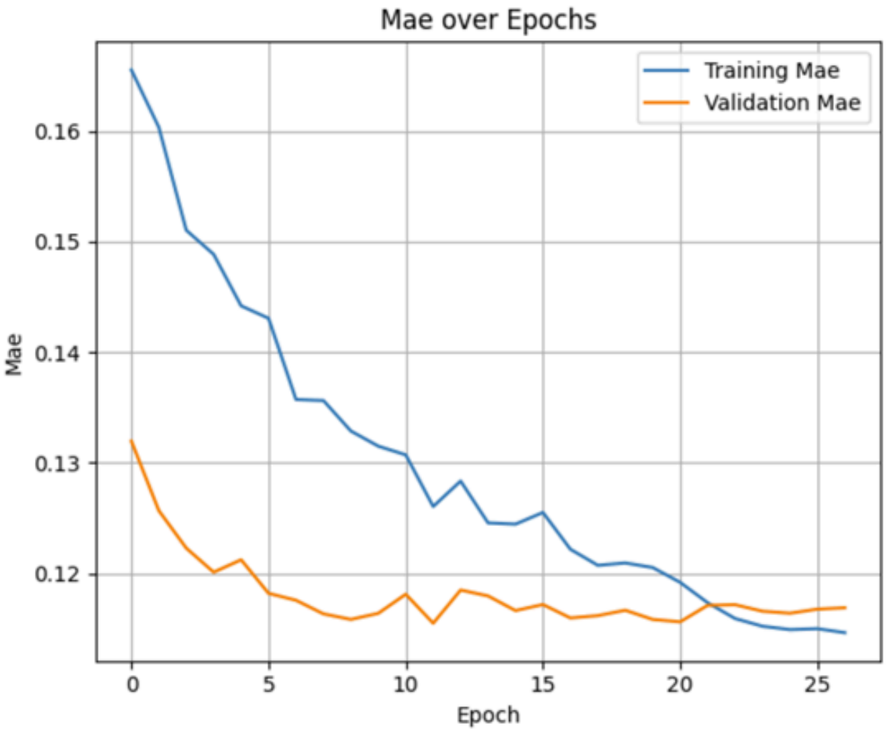
WITH DATA-AUGMENTATION

```python
import albumentations as A
from albumentations.core.composition import Compose, OneOf
from albumentations.pytorch import ToTensorV2
import numpy as np
import tensorflow as tf

# Define the Albumentations augmentation pipeline
augmentation_pipeline = A.Compose([
    A.Rotate(limit=10, p=0.5),                  # Rotation range of 10 degrees
    A.HorizontalFlip(p=0.5),                    # Horizontal flip
    A.ShiftScaleRotate(shift_limit=0.1, scale_limit=0.1, rotate_limit=0, p=0.5),
    A.RandomBrightnessContrast(brightness_limit=0.1, contrast_limit=0.1, p=0.5),
#    A.RandomZoom(scale_limit=0.1, p=0.5),
    A.Normalize(),                              # Normalization
])

# Custom data generator to apply Albumentations on-the-fly
class AlbumentationsDataGenerator(tf.keras.utils.Sequence):
    def __init__(self, images, labels, batch_size=32, augmentations=None, shuffle=True):
        self.images = images
        self.labels = labels
        self.batch_size = batch_size
        self.augmentations = augmentations
        self.shuffle = shuffle
        self.indices = np.arange(len(self.images))
        self.on_epoch_end()

    def __len__(self):
         return len(self.images) // self.batch_size
```

```
    def __getitem__(self, index):
        batch_indices = self.indices[index * self.batch_size:(index + 1) * self.batch_size]
        batch_images = [self.images[i] for i in batch_indices]
        batch_labels = [self.labels[i] for i in batch_indices]

        # Apply augmentations
        if self.augmentations:
            batch_images = [self.augmentations(image=image)['image'] for image in batch_images]
        return np.array(batch_images), np.array(batch_labels)

    def on_epoch_end(self):
        if self.shuffle:
            np.random.shuffle(self.indices)

# Initialize the custom data generator with augmentations
train_generator = AlbumentationsDataGenerator(
    images=X_train,
    labels=Y_train,
    batch_size=32,
    augmentations=augmentation_pipeline
)
Nasnet_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5), loss='mse', metrics=['mae'])

# Training the model
history = Nasnet_model.fit(
    train_generator,
    validation_data=(X_valid_processed_Nasnet, Y_valid),
    epochs=100,
    shuffle=True,
    verbose=1,
    callbacks=[early_stopping_Nasnet]
)
```

```
Epoch 12/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 13s 98ms/step - loss: 0.0269 - mae: 0.1285 - val_loss: 0.0209 - val_mae: 0.1155
Epoch 13/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 20s 95ms/step - loss: 0.0264 - mae: 0.1272 - val_loss: 0.0215 - val_mae: 0.1185
Epoch 14/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 13s 102ms/step - loss: 0.0247 - mae: 0.1237 - val_loss: 0.0214 - val_mae: 0.1180
Epoch 15/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 12s 92ms/step - loss: 0.0261 - mae: 0.1259 - val_loss: 0.0210 - val_mae: 0.1166
Epoch 16/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 12s 94ms/step - loss: 0.0260 - mae: 0.1271 - val_loss: 0.0212 - val_mae: 0.1172
Epoch 17/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 11s 86ms/step - loss: 0.0242 - mae: 0.1209 - val_loss: 0.0209 - val_mae: 0.1160
Epoch 18/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 12s 92ms/step - loss: 0.0235 - mae: 0.1199 - val_loss: 0.0209 - val_mae: 0.1162
Epoch 19/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 11s 88ms/step - loss: 0.0236 - mae: 0.1209 - val_loss: 0.0210 - val_mae: 0.1167
Epoch 20/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 11s 83ms/step - loss: 0.0236 - mae: 0.1204 - val_loss: 0.0208 - val_mae: 0.1158
Epoch 21/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 90ms/step - loss: 0.0235 - mae: 0.1202 - val_loss: 0.0207 - val_mae: 0.1156
Epoch 22/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 11s 88ms/step - loss: 0.0237 - mae: 0.1188 - val_loss: 0.0210 - val_mae: 0.1171
Epoch 23/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 99ms/step - loss: 0.0220 - mae: 0.1165 - val_loss: 0.0210 - val_mae: 0.1172
Epoch 24/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 12s 96ms/step - loss: 0.0218 - mae: 0.1155 - val_loss: 0.0209 - val_mae: 0.1166
Epoch 25/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 20s 86ms/step - loss: 0.0219 - mae: 0.1159 - val_loss: 0.0208 - val_mae: 0.1164
Epoch 26/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 90ms/step - loss: 0.0222 - mae: 0.1170 - val_loss: 0.0209 - val_mae: 0.1168
Epoch 27/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 91ms/step - loss: 0.0208 - mae: 0.1137 - val_loss: 0.0210 - val_mae: 0.1169
Epoch 27: early stopping
```

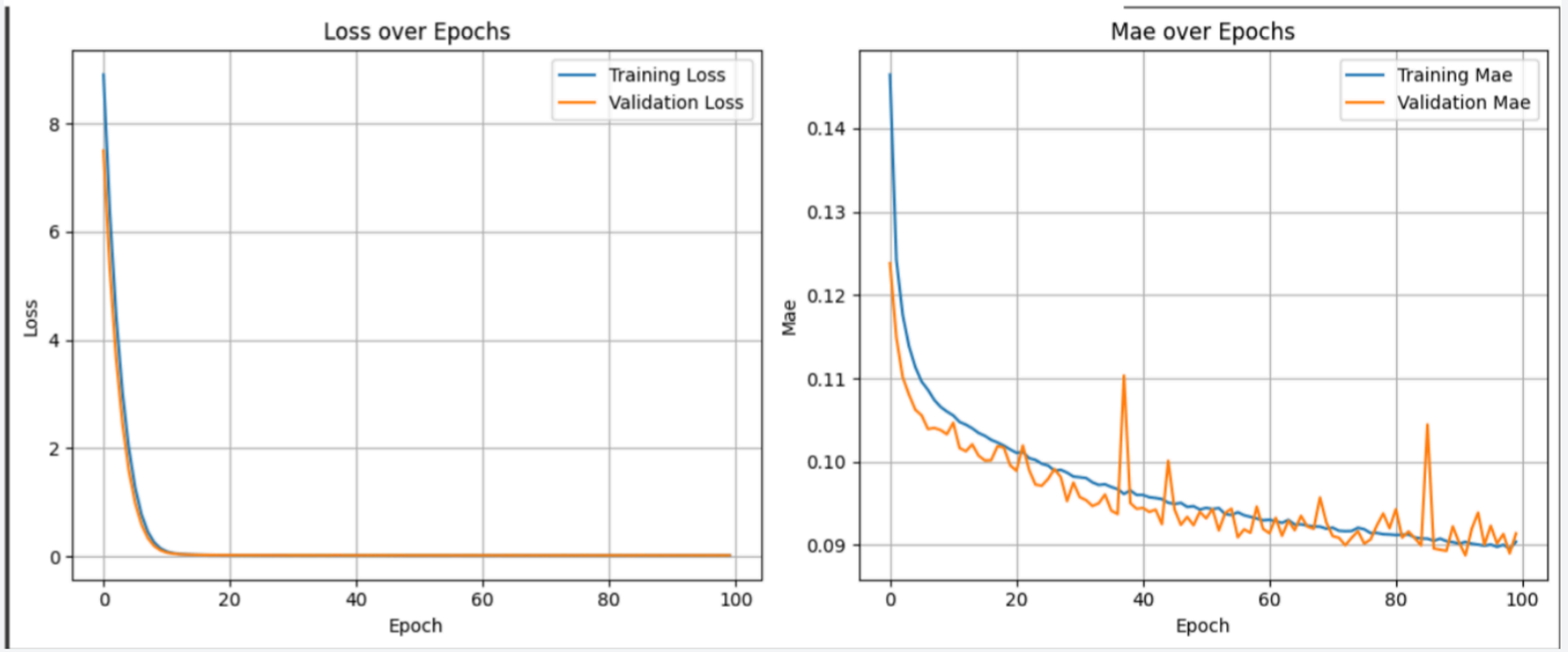




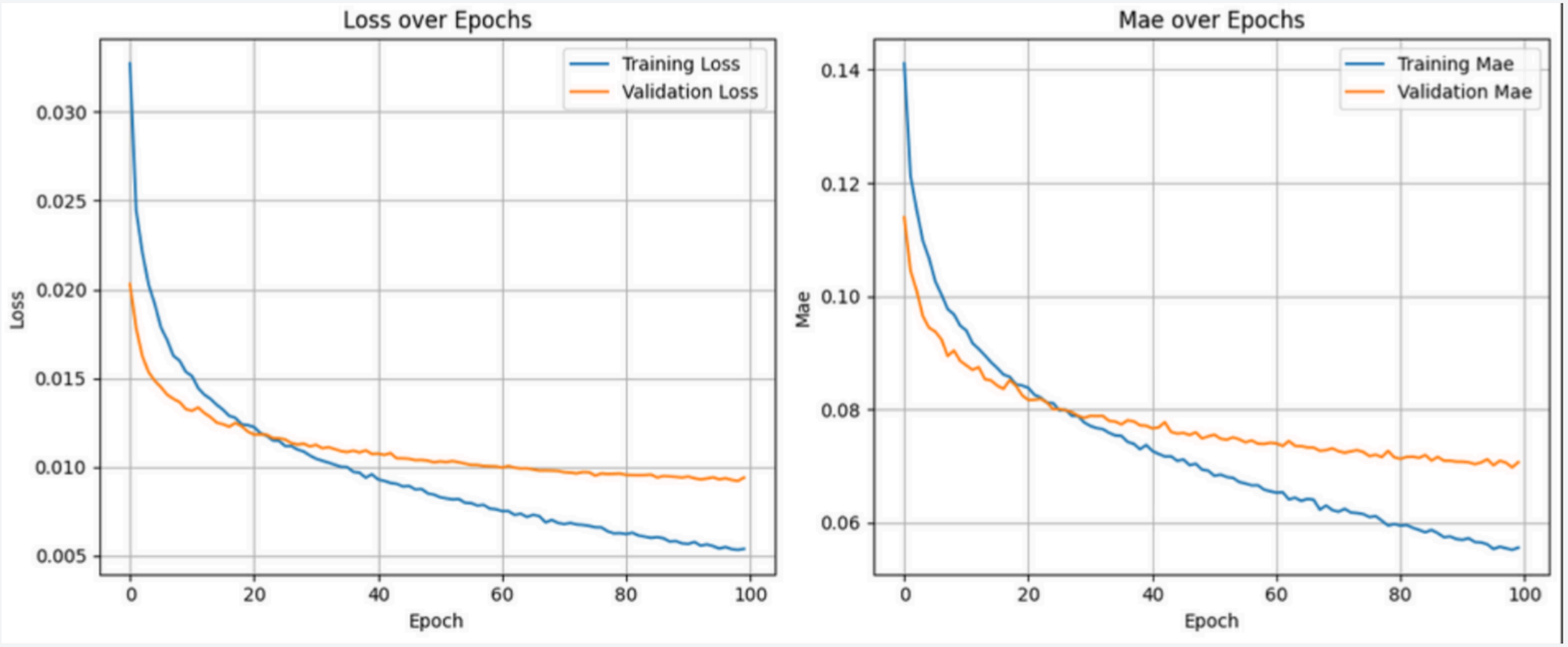

```
Current learning rate: 1e-05
```

```python
### Nasnet implementation avec 0.7 de Dropout and with 1 trainable layer
early_stopping_Nasnet = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=6)
# Load NASNetMobile with pre-trained ImageNet weights, excluding the top layer
base_model = tf.keras.applications.NASNetMobile(input_shape=(224, 224, 3), weights='imagenet', include_top=False)
#preprocess_input = tf.keras.applications.mobilenet.preprocess_input
preprocess_input_Nasnet = tf.keras.applications.nasnet.preprocess_input
base_model.trainable = False
for layer in base_model.layers[-3:]:
    layer.trainable = True
# Adding custom layers
x = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
x = tf.keras.layers.Dense(1024, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01))(x)
#x = tf.keras.layers.Dropout(0.5)(x)  # Dropout with 50% probability
output = tf.keras.layers.Dense(1, activation='sigmoid')(x)
# Create the new model with NASNetMobile as the base
Nasnet_model = tf.keras.models.Model(inputs=base_model.input, outputs=output)
# Preprocess the data
X_train_processed_Nasnet = preprocess_input_Nasnet(np.copy(X_train2))
X_valid_processed_Nasnet = preprocess_input_Nasnet(np.copy(X_valid2))
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
print("X_valid_processed_Nasnet shape:", X_valid_processed_Nasnet.shape)
print("Y_valid2 shape:", Y_valid2.shape)
```

4x more examples : 20K


Loss over Epochs
Training Loss
Validation Loss
Loss
Epoch
0
2
4
6
8
0
20
40
60
80
100
Mae over Epochs
Training Mae
Validation Mae
Mae
Epoch
0.09
0.10
0.11
0.12
0.13
0.14
0
20
40
60
80
100

```python
### Mobilenet basic
early_stopping_MobileNet = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=7)

#base_model = tf.keras.applications.ResNet50(input_shape=(224, 224, 3), weights='imagenet', include_top=False)
base_model = tf.keras.applications.MobileNetV3Small(input_shape=(224, 224, 3), weights='imagenet', include_top=False)

#preprocess_input = tf.keras.applications.mobilenet.preprocess_input
preprocess_input_Mobilenet = tf.keras.applications.mobilenet_v3.preprocess_input

base_model.trainable = False


base_model.trainable = False
for layer in base_model.layers[-3:]:
    layer.trainable = True

# Add custom layers
x = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
x = tf.keras.layers.Dense(1024, activation='relu')(x)
# = tf.keras.layers.Dropout(0.5)(x)
output = tf.keras.layers.Dense(1, activation='sigmoid')(x)

# Create the new model with MobileNet as the base
Mobilenet_model = tf.keras.models.Model(inputs=base_model.input, outputs=output)
# Preprocess the data
X_train_processed_Mobilenet = preprocess_input_Mobilenet(np.copy(X_train2))
X_valid_processed_Mobilenet = preprocess_input_Mobilenet(np.copy(X_valid2))

print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))

print(f'Nb layer : {len(Mobilenet_model.layers)}')
```


Loss over Epochs
Training Loss
Validation Loss
Loss
Epoch
0.030
0.025
0.020
0.015
0.010
0.005
0
20
40
60
80
100
Mae over Epochs
Training Mae
Validation Mae
Mae
Epoch
0.14
0.12
0.10
0.08
0.06

Train Model

MODEL ENSEMBLING

```
[81] from sklearn.metrics import mean_absolute_error
     Nasnet_prediction = Nasnet_model.predict(X_valid2)
     Mobilenet_prediction = Mobilenet_model.predict(X_valid2)
     predictions = (Nasnet_prediction*0.4 ) + (0.6*Mobilenet_prediction)
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
```

```
mae = mean_absolute_error(Y_valid2, predictions)
print(f"Mean Absolute Error (MAE): {mae}")
```

```
Mean Absolute Error (MAE): 0.8256474798896908
```

Train Model

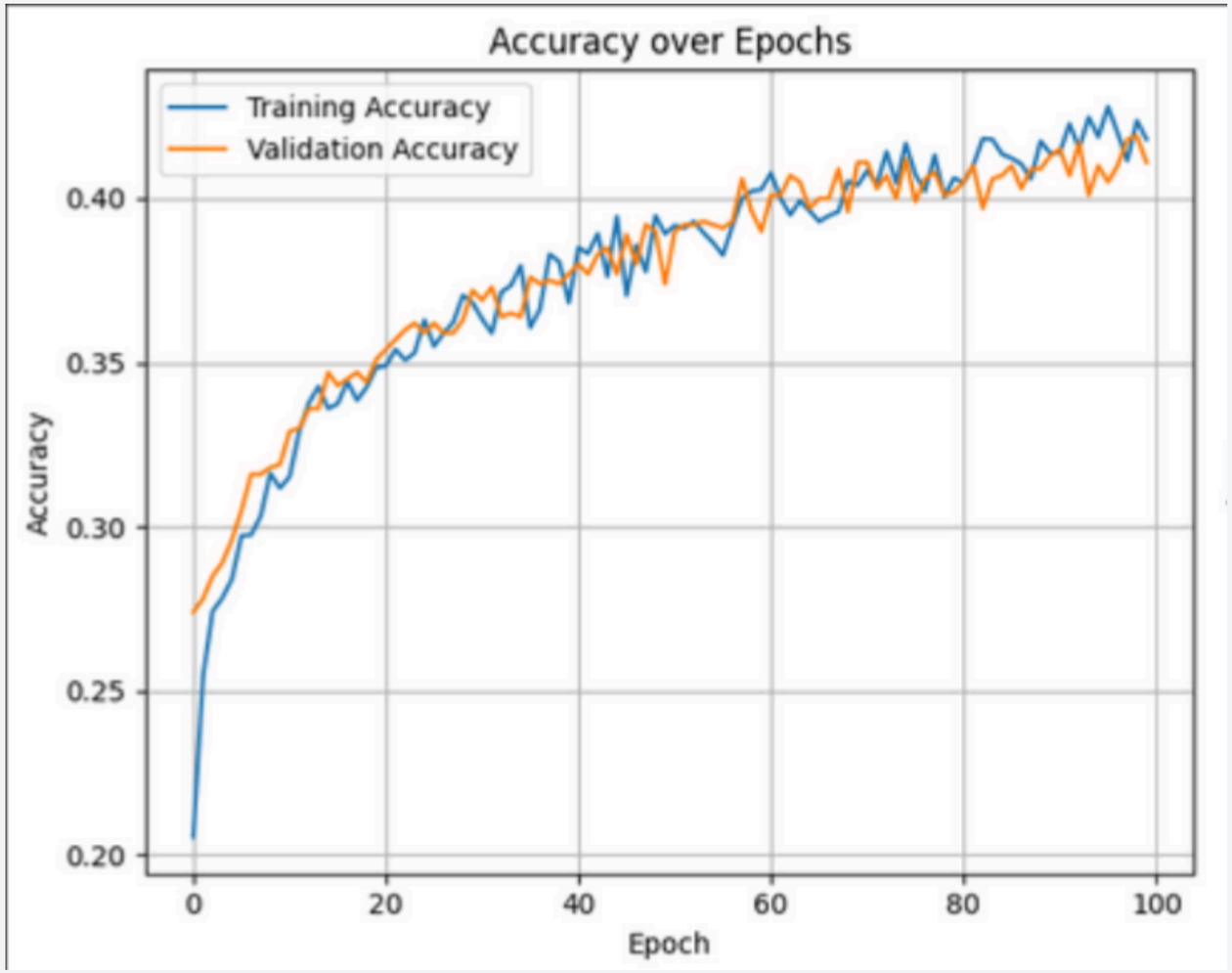
CLASSIFICATION

Age Buckets

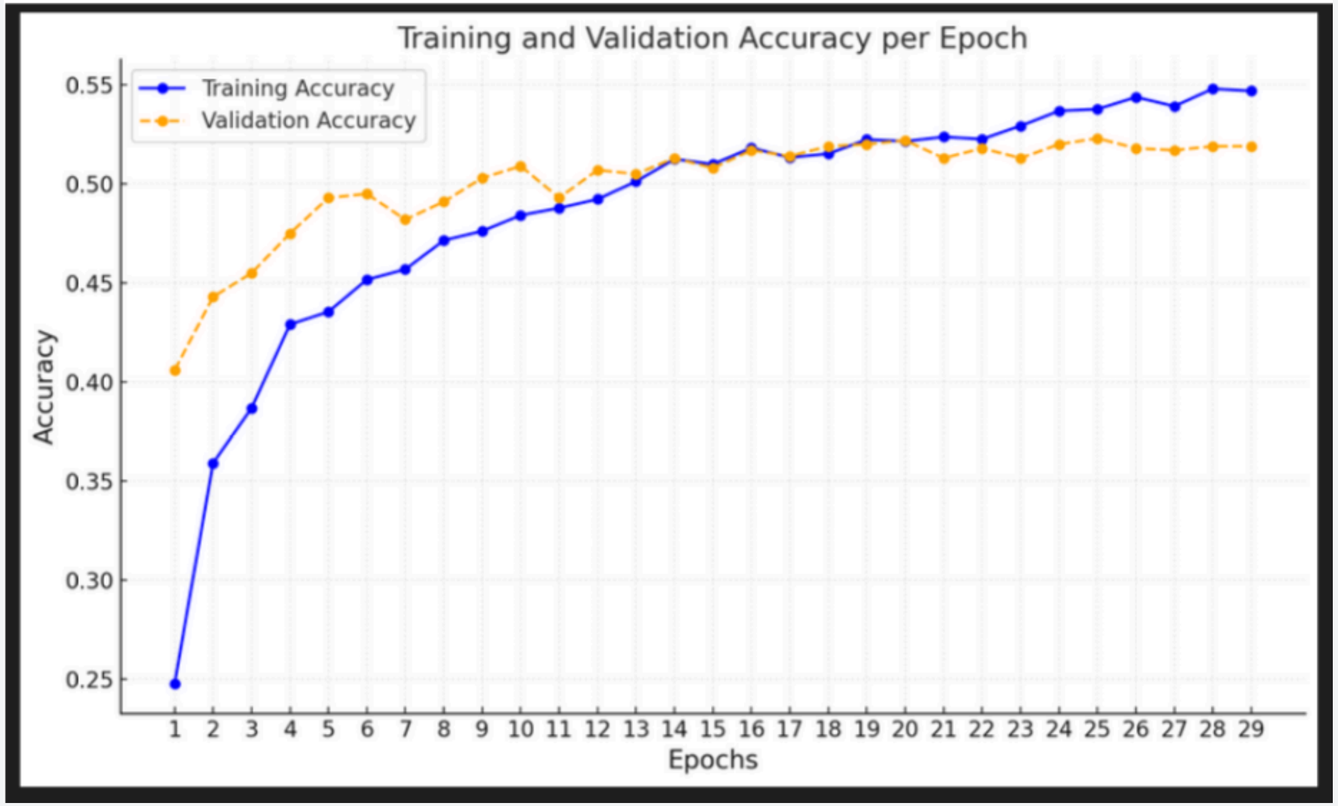
```
[13] # defining category

     def age_to_class(age):
         age = int(age)
         if 0 <= age <= 2:
             return 0
         elif 3 <= age <= 9:
             return 1
         elif 10 <= age <= 20:
             return 2
         elif 21 <= age <= 27:
             return 3
         elif 28 <= age <= 45:
             return 4
         elif 46 <= age <= 65:
             return 5
         else:
           return 6
```

1 sigmoid -> 7 softmax

NasNet


Accuracy over Epochs
Training Accuracy
Validation Accuracy
Accuracy
0.40
0.35
0.30
0.25
0.20
0
20
40
60
80
100
Epoch

MobileNet V3 Small



Comparison

with

Production Level Models

Model	Dataset	MAE (Years)	Accuracy (%)
VGGNet	IMDB-WIKI	~4.2–4.8	~75–80%
	FG-NET	~3.1–3.5	~80–85%
ResNet	IMDB-WIKI	~3.2–3.8	~82–88%
	FG-NET	~2.8–3.0	~85–90%
	APPA-REAL	~3.5–4.0	~78–83%
EfficientNet	IMDB-WIKI	~2.8–3.5	~85–90%
	FG-NET	~2.5–2.8	~88–92%
	APPA-REAL	~3.2–3.8	~82–87%
Vision Transformers (ViT)	IMDB-WIKI	~2.6–3.2	~87–92%
	FG-NET	~2.4–2.7	~90–94%
	APPA-REAL	~3.0–3.5	~85–90%

Nasnet (TL)

val_mae: ~9

accuracy: ~45%

MobileNet (TL)

val_mae: ~7

accuracy: ~55%

Thank you!