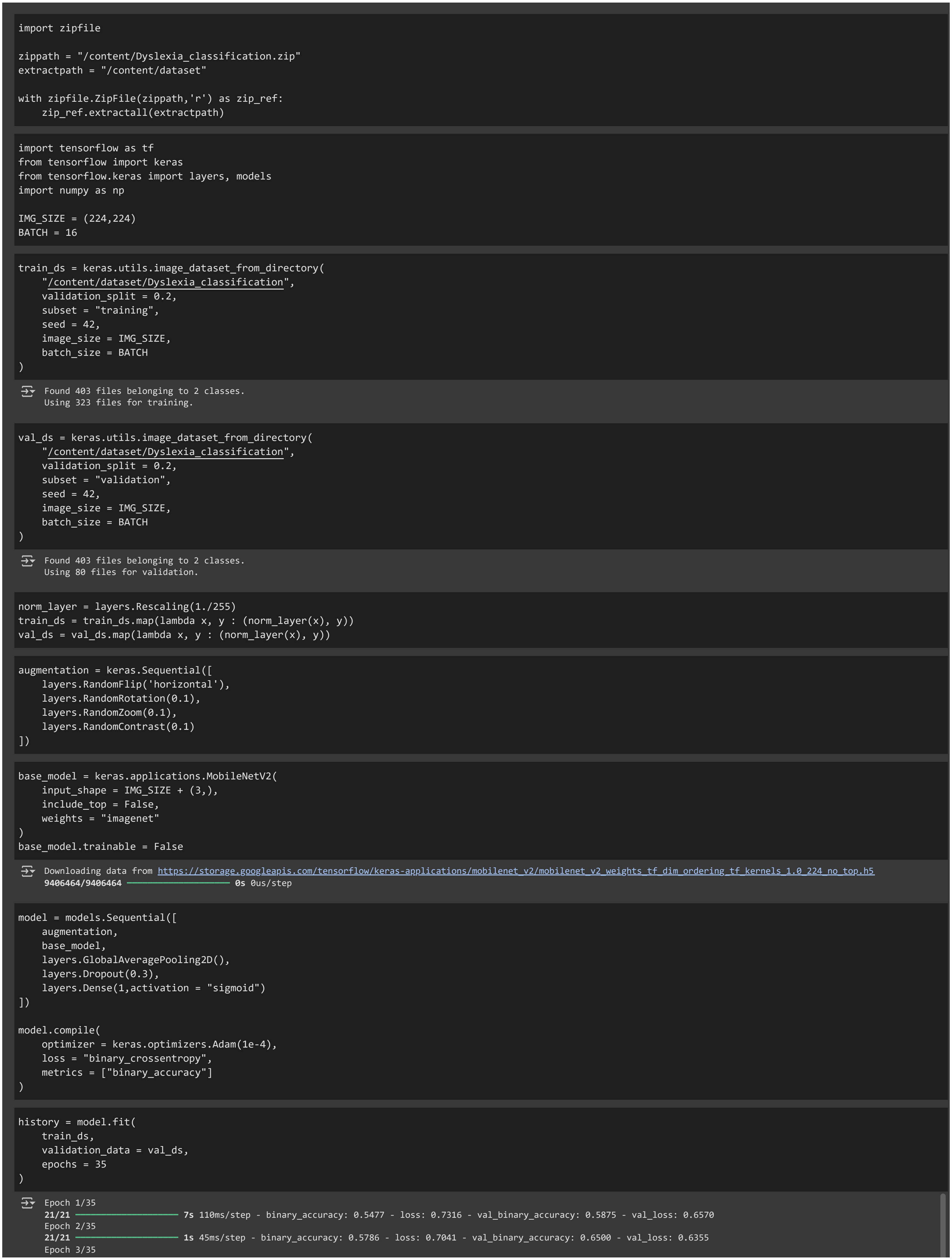


```
import zipfile

zippath = "/content/Dyslexia_classification.zip"
extractpath = "/content/dataset"

with zipfile.ZipFile(zippath,'r') as zip_ref:
    zip_ref.extractall(extractpath)
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
import numpy as np

IMG_SIZE = (224,224)
BATCH = 16
```

```
train_ds = keras.utils.image_dataset_from_directory(
    "/content/dataset/Dyslexia_classification",
    validation_split = 0.2,
    subset = "training",
    seed = 42,
    image_size = IMG_SIZE,
    batch_size = BATCH
)
```

```
Found 403 files belonging to 2 classes.
Using 323 files for training.
```

```
val_ds = keras.utils.image_dataset_from_directory(
    "/content/dataset/Dyslexia_classification",
    validation_split = 0.2,
    subset = "validation",
    seed = 42,
    image_size = IMG_SIZE,
    batch_size = BATCH
)
```

```
Found 403 files belonging to 2 classes.
Using 80 files for validation.
```

```
norm_layer = layers.Rescaling(1./255)
train_ds = train_ds.map(lambda x, y : (norm_layer(x), y))
val_ds = val_ds.map(lambda x, y : (norm_layer(x), y))
```

```
augmentation = keras.Sequential([
    layers.RandomFlip('horizontal'),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
    layers.RandomContrast(0.1)
])
```

```
base_model = keras.applications.MobileNetV2(
    input_shape = IMG_SIZE + (3,),
    include_top = False,
    weights = "imagenet"
)
base_model.trainable = False
```

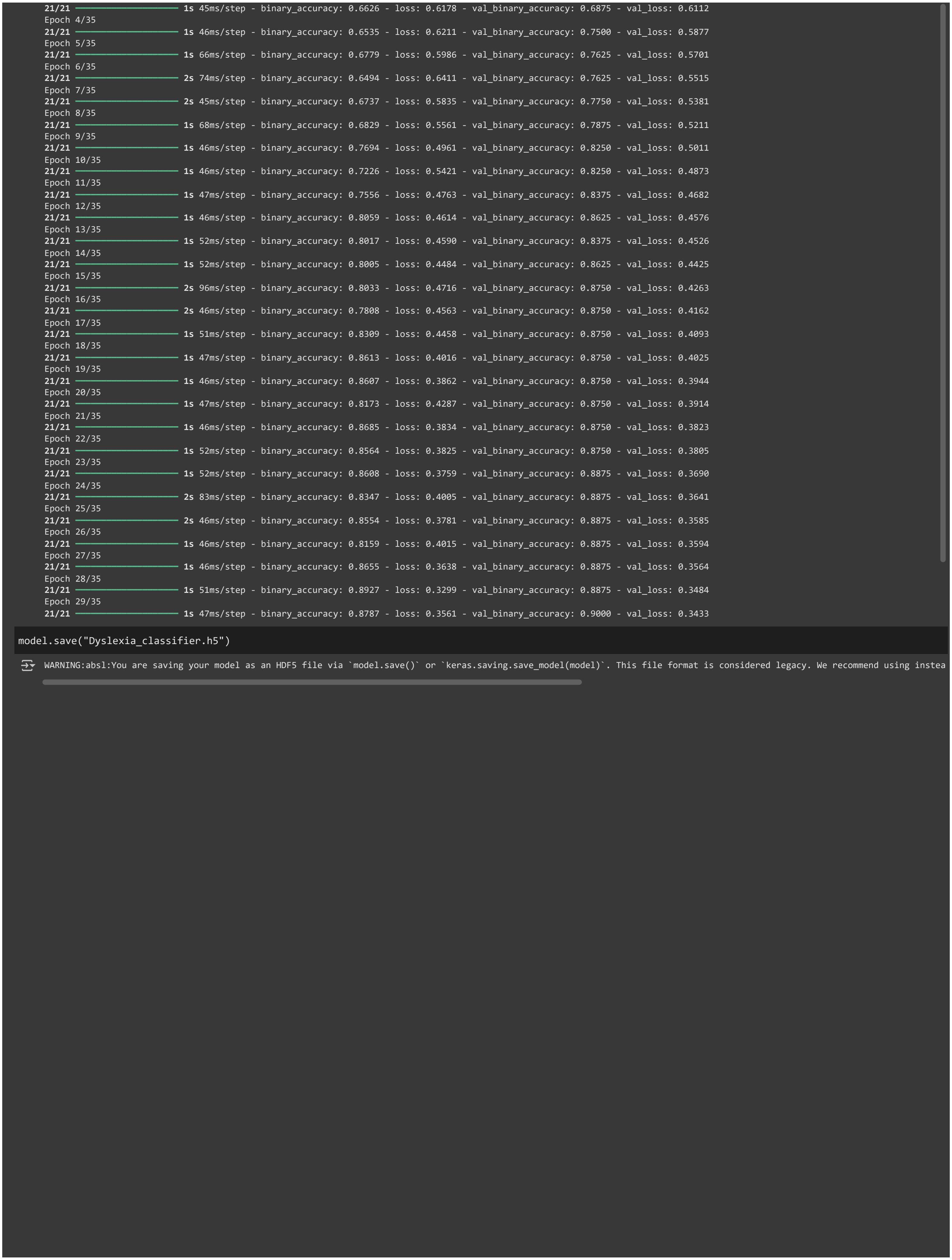
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
model = models.Sequential([
    augmentation,
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dropout(0.3),
    layers.Dense(1,activation = "sigmoid")
])

model.compile(
    optimizer = keras.optimizers.Adam(1e-4),
    loss = "binary_crossentropy",
    metrics = ["binary_accuracy"]
)
```

```
history = model.fit(
    train_ds,
    validation_data = val_ds,
    epochs = 35
)
```

```
Epoch 1/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 7s 110ms/step - binary_accuracy: 0.5477 - loss: 0.7316 - val_binary_accuracy: 0.5875 - val_loss: 0.6570
Epoch 2/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - binary_accuracy: 0.5786 - loss: 0.7041 - val_binary_accuracy: 0.6500 - val_loss: 0.6355
Epoch 3/35
```



```
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - binary_accuracy: 0.6626 - loss: 0.6178 - val_binary_accuracy: 0.6875 - val_loss: 0.6112
Epoch 4/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - binary_accuracy: 0.6535 - loss: 0.6211 - val_binary_accuracy: 0.7500 - val_loss: 0.5877
Epoch 5/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 66ms/step - binary_accuracy: 0.6779 - loss: 0.5986 - val_binary_accuracy: 0.7625 - val_loss: 0.5701
Epoch 6/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 2s 74ms/step - binary_accuracy: 0.6494 - loss: 0.6411 - val_binary_accuracy: 0.7625 - val_loss: 0.5515
Epoch 7/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 2s 45ms/step - binary_accuracy: 0.6737 - loss: 0.5835 - val_binary_accuracy: 0.7750 - val_loss: 0.5381
Epoch 8/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 68ms/step - binary_accuracy: 0.6829 - loss: 0.5561 - val_binary_accuracy: 0.7875 - val_loss: 0.5211
Epoch 9/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - binary_accuracy: 0.7694 - loss: 0.4961 - val_binary_accuracy: 0.8250 - val_loss: 0.5011
Epoch 10/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - binary_accuracy: 0.7226 - loss: 0.5421 - val_binary_accuracy: 0.8250 - val_loss: 0.4873
Epoch 11/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - binary_accuracy: 0.7556 - loss: 0.4763 - val_binary_accuracy: 0.8375 - val_loss: 0.4682
Epoch 12/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - binary_accuracy: 0.8059 - loss: 0.4614 - val_binary_accuracy: 0.8625 - val_loss: 0.4576
Epoch 13/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 52ms/step - binary_accuracy: 0.8017 - loss: 0.4590 - val_binary_accuracy: 0.8375 - val_loss: 0.4526
Epoch 14/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 52ms/step - binary_accuracy: 0.8005 - loss: 0.4484 - val_binary_accuracy: 0.8625 - val_loss: 0.4425
Epoch 15/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 2s 96ms/step - binary_accuracy: 0.8033 - loss: 0.4716 - val_binary_accuracy: 0.8750 - val_loss: 0.4263
Epoch 16/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 2s 46ms/step - binary_accuracy: 0.7808 - loss: 0.4563 - val_binary_accuracy: 0.8750 - val_loss: 0.4162
Epoch 17/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 51ms/step - binary_accuracy: 0.8309 - loss: 0.4458 - val_binary_accuracy: 0.8750 - val_loss: 0.4093
Epoch 18/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - binary_accuracy: 0.8613 - loss: 0.4016 - val_binary_accuracy: 0.8750 - val_loss: 0.4025
Epoch 19/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - binary_accuracy: 0.8607 - loss: 0.3862 - val_binary_accuracy: 0.8750 - val_loss: 0.3944
Epoch 20/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - binary_accuracy: 0.8173 - loss: 0.4287 - val_binary_accuracy: 0.8750 - val_loss: 0.3914
Epoch 21/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - binary_accuracy: 0.8685 - loss: 0.3834 - val_binary_accuracy: 0.8750 - val_loss: 0.3823
Epoch 22/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 52ms/step - binary_accuracy: 0.8564 - loss: 0.3825 - val_binary_accuracy: 0.8750 - val_loss: 0.3805
Epoch 23/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 52ms/step - binary_accuracy: 0.8608 - loss: 0.3759 - val_binary_accuracy: 0.8875 - val_loss: 0.3690
Epoch 24/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 2s 83ms/step - binary_accuracy: 0.8347 - loss: 0.4005 - val_binary_accuracy: 0.8875 - val_loss: 0.3641
Epoch 25/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 2s 46ms/step - binary_accuracy: 0.8554 - loss: 0.3781 - val_binary_accuracy: 0.8875 - val_loss: 0.3585
Epoch 26/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - binary_accuracy: 0.8159 - loss: 0.4015 - val_binary_accuracy: 0.8875 - val_loss: 0.3594
Epoch 27/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - binary_accuracy: 0.8655 - loss: 0.3638 - val_binary_accuracy: 0.8875 - val_loss: 0.3564
Epoch 28/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 51ms/step - binary_accuracy: 0.8927 - loss: 0.3299 - val_binary_accuracy: 0.8875 - val_loss: 0.3484
Epoch 29/35
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - binary_accuracy: 0.8787 - loss: 0.3561 - val_binary_accuracy: 0.9000 - val_loss: 0.3433
```

```
model.save("Dyslexia_classifier.h5")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instea
```