```
# === KAGGLE ===
!pip install kaggle
```

Show hidden output

```
# === Upload Kaggle.json ===
from google.colab import files
files.upload()
```

Choose Files kaggle.json

- **kaggle.json**(application/json) - 65 bytes, last modified: 4/22/2025 - 100% done

```
Saving kaggle.json to kaggle.json
```

```
# === Setup Kaggle API ===
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
# === Downloading the dataset ===
!kaggle datasets download drscarlat/melanoma
```

```
Dataset URL: https://www.kaggle.com/datasets/drscarlat/melanoma
License(s): unknown
```

```
# === Unzipping the dataset ===
!unzip melanoma.zip
```

Show hidden output

```
# === Import necessary libraries ===
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50, EfficientNetB0
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout, BatchNormalization, GaussianNoise, ELU
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
```

```
# === Set image dimensions and batch size ===
img_height, img_width = 224, 224
batch_size = 32
```

```
# === Data augmentation for training and validation ===
train_aug = ImageDataGenerator(
    rescale=1.0 / 255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.2
)

val_aug = ImageDataGenerator(
    rescale=1.0 / 255,
    validation_split=0.2
)
```

```
# === Load training and validation data ===
train_data = train_aug.flow_from_directory(
    '/content/DermMel',
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

val_data = val_aug.flow_from_directory(
    '/content/dermmel/DermMel',
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 14245 images belonging to 3 classes.
Found 3560 images belonging to 3 classes.
```

```
# === Build ResNet50 model ===
def build_resnet_model(input_shape, num_classes):
    base_model = ResNet50(weights='imagenet', include_top=False, input_shape=input_shape)
    for layer in base_model.layers[:100]:
        layer.trainable = False
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = BatchNormalization()(x)
    x = GaussianNoise(0.1)(x)
    x = Dense(512)(x)
    x = ELU()(x)
    x = Dropout(0.5)(x)
    output = Dense(num_classes, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=output)
    return model
```

```
# === Build EfficientNetB0 model ===
def build_efficientnet_model(input_shape, num_classes):
    base_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=input_shape)
    for layer in base_model.layers[:len(base_model.layers) // 2]:
        layer.trainable = False
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = BatchNormalization()(x)
    x = GaussianNoise(0.15)(x)
    x = Dense(256)(x)
    x = ELU()(x)
    x = Dropout(0.5)(x)
    output = Dense(num_classes, activation='softmax')(x)
```

```
    model = Model(inputs=base_model.input, outputs=output)
    return model
```

```
# === Callbacks ===
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True, verbose=1)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, verbose=1, min_lr=1e-6)
checkpoint_resnet = ModelCheckpoint('best_resnet_model.h5', monitor='val_loss', save_best_only=True, verbose=1)
checkpoint_efficientnet = ModelCheckpoint('best_efficientnet_model.h5', monitor='val_loss', save_best_only=True, verbose=1)
```

```
# === Train ResNet50 ===
resnet_model = build_resnet_model((img_height, img_width, 3), train_data.num_classes)
resnet_model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
resnet_history = resnet_model.fit(train_data, validation_data=val_data, epochs=20, callbacks=[early_stopping, reduce_lr, checl
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This fi
446/446 ━━━━━━━━━━━━━━━━━━━━ 254s 569ms/step - accuracy: 0.5826 - loss: 0.9971 - val_accuracy: 0.5997 - val_loss: 1.572
Epoch 3/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 541ms/step - accuracy: 0.5865 - loss: 0.9908
Epoch 3: val_loss did not improve from 1.57242
446/446 ━━━━━━━━━━━━━━━━━━━━ 251s 562ms/step - accuracy: 0.5865 - loss: 0.9907 - val_accuracy: 0.6000 - val_loss: 2.267
Epoch 4/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 541ms/step - accuracy: 0.5981 - loss: 0.9744
```

```
Epoch 7: val_loss did not improve from 1.03770
446/446 ━━━━━━━━━━━━━━━━━━━━ 250s 561ms/step - accuracy: 0.5985 - loss: 0.9664 - val_accuracy: 0.2000 - val_loss: 4.389
Epoch 8/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 545ms/step - accuracy: 0.5978 - loss: 0.9680
Epoch 8: val_loss did not improve from 1.03770
446/446 ━━━━━━━━━━━━━━━━━━━━ 264s 566ms/step - accuracy: 0.5978 - loss: 0.9680 - val_accuracy: 0.2000 - val_loss: 8.492
Epoch 9/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 538ms/step - accuracy: 0.5952 - loss: 0.9705
Epoch 9: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.

Epoch 9: val_loss did not improve from 1.03770
446/446 ━━━━━━━━━━━━━━━━━━━━ 250s 559ms/step - accuracy: 0.5952 - loss: 0.9705 - val_accuracy: 0.6000 - val_loss: 2.844
Epoch 10/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 533ms/step - accuracy: 0.6031 - loss: 0.9613
Epoch 10: val_loss did not improve from 1.03770
446/446 ━━━━━━━━━━━━━━━━━━━━ 247s 554ms/step - accuracy: 0.6031 - loss: 0.9613 - val_accuracy: 0.2000 - val_loss: 3.986
Epoch 11/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 534ms/step - accuracy: 0.6041 - loss: 0.9580
Epoch 11: val_loss did not improve from 1.03770
446/446 ━━━━━━━━━━━━━━━━━━━━ 248s 556ms/step - accuracy: 0.6041 - loss: 0.9580 - val_accuracy: 0.2000 - val_loss: 5.122
Epoch 12/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 534ms/step - accuracy: 0.6017 - loss: 0.9585
Epoch 12: val_loss did not improve from 1.03770
446/446 ━━━━━━━━━━━━━━━━━━━━ 247s 555ms/step - accuracy: 0.6017 - loss: 0.9585 - val_accuracy: 0.6000 - val_loss: 2.853
Epoch 13/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 533ms/step - accuracy: 0.6027 - loss: 0.9589
Epoch 13: val_loss did not improve from 1.03770
446/446 ━━━━━━━━━━━━━━━━━━━━ 247s 555ms/step - accuracy: 0.6027 - loss: 0.9589 - val_accuracy: 0.6000 - val_loss: 2.158
Epoch 14/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 536ms/step - accuracy: 0.5946 - loss: 0.9652
Epoch 14: ReduceLROnPlateau reducing learning rate to 2.499999936844688e-05.

Epoch 14: val_loss did not improve from 1.03770
446/446 ━━━━━━━━━━━━━━━━━━━━ 249s 558ms/step - accuracy: 0.5947 - loss: 0.9652 - val_accuracy: 0.2000 - val_loss: 8.671
Epoch 14: early stopping
Restoring model weights from the end of the best epoch: 4.
```

```
# === Train EfficientNetB0 ===
efficientnet_model = build_efficientnet_model((img_height, img_width, 3), train_data.num_classes)
efficientnet_model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
efficientnet_history = efficientnet_model.fit(train_data, validation_data=val_data, epochs=20, callbacks=[early_stopping, reduc
```

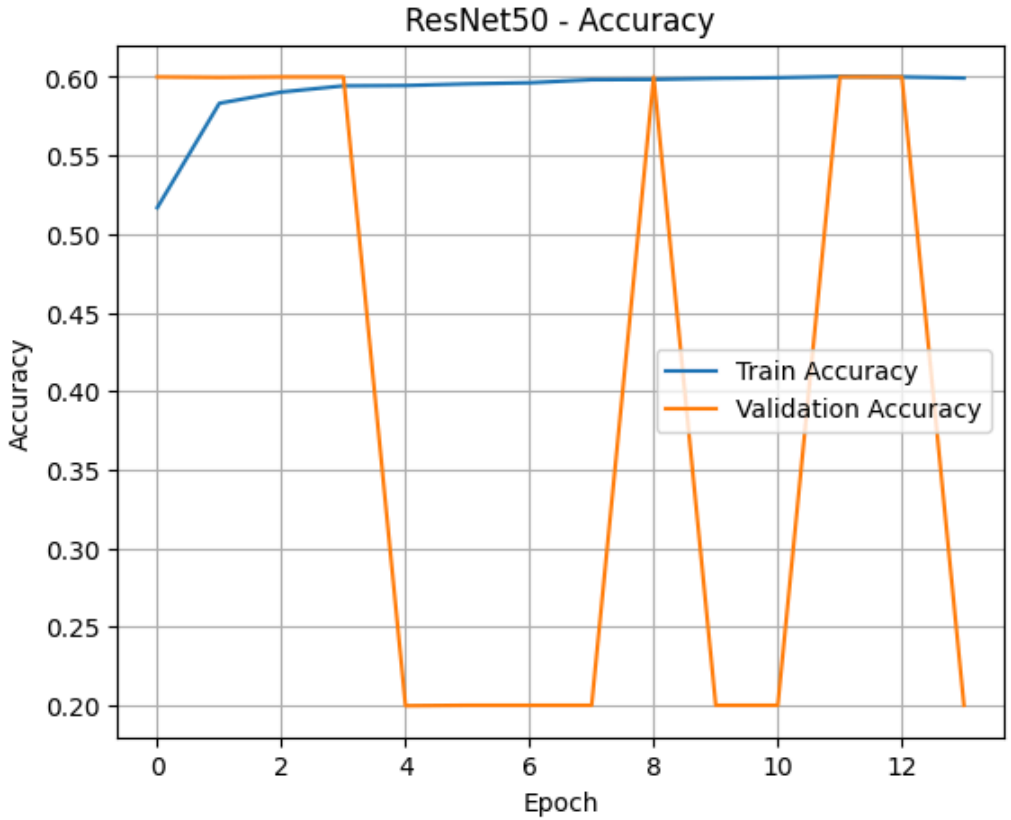
```
446/446 ━━━━━━━━━━━━━━━━━━━━ 234s 525ms/step - accuracy: 0.5589 - loss: 1.0522 - val_accuracy: 0.6000 - val_loss: 0.96
Epoch 4/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 511ms/step - accuracy: 0.5679 - loss: 1.0362
Epoch 4: val_loss improved from 0.95998 to 0.95386, saving model to best_efficientnet_model.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This fil
446/446 ━━━━━━━━━━━━━━━━━━━━ 261s 524ms/step - accuracy: 0.5679 - loss: 1.0362 - val_accuracy: 0.6000 - val_loss: 0.953
Epoch 5/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 522ms/step - accuracy: 0.5806 - loss: 1.0199
Epoch 5: val_loss improved from 0.95386 to 0.95141, saving model to best_efficientnet_model.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This fil
446/446 ━━━━━━━━━━━━━━━━━━━━ 238s 534ms/step - accuracy: 0.5806 - loss: 1.0199 - val_accuracy: 0.6000 - val_loss: 0.951
Epoch 6/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 515ms/step - accuracy: 0.5808 - loss: 1.0106
Epoch 6: val_loss did not improve from 0.95141
446/446 ━━━━━━━━━━━━━━━━━━━━ 235s 526ms/step - accuracy: 0.5808 - loss: 1.0106 - val_accuracy: 0.6000 - val_loss: 0.952
Epoch 7/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 517ms/step - accuracy: 0.5896 - loss: 1.0040
```

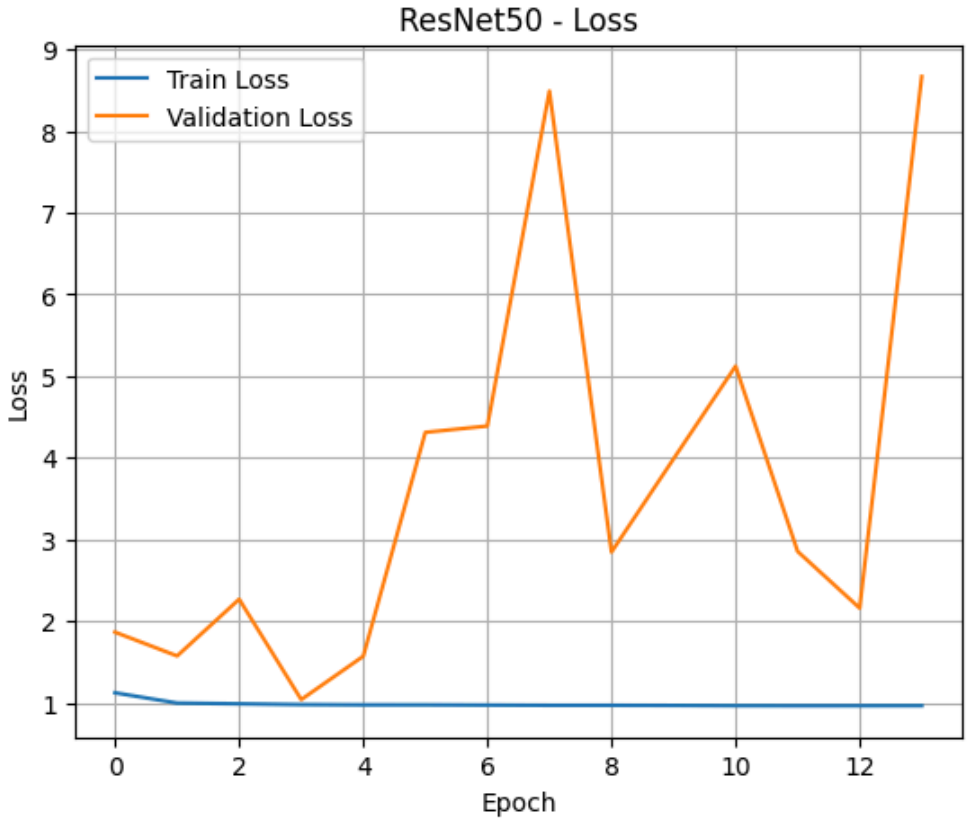
```
Epoch 11/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 523ms/step - accuracy: 0.5855 - loss: 0.9926
Epoch 11: val_loss did not improve from 0.95141
446/446 ━━━━━━━━━━━━━━━━━━━━ 238s 534ms/step - accuracy: 0.5855 - loss: 0.9926 - val_accuracy: 0.2697 - val_loss: 2.995
Epoch 12/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 516ms/step - accuracy: 0.5977 - loss: 0.9801
Epoch 12: val_loss did not improve from 0.95141
446/446 ━━━━━━━━━━━━━━━━━━━━ 235s 527ms/step - accuracy: 0.5977 - loss: 0.9801 - val_accuracy: 0.2171 - val_loss: 7.331
Epoch 13/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 521ms/step - accuracy: 0.6011 - loss: 0.9700
Epoch 13: val_loss did not improve from 0.95141
446/446 ━━━━━━━━━━━━━━━━━━━━ 238s 533ms/step - accuracy: 0.6011 - loss: 0.9700 - val_accuracy: 0.2180 - val_loss: 4.598
Epoch 14/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 519ms/step - accuracy: 0.5911 - loss: 0.9786
Epoch 14: val_loss did not improve from 0.95141
446/446 ━━━━━━━━━━━━━━━━━━━━ 242s 542ms/step - accuracy: 0.5911 - loss: 0.9785 - val_accuracy: 0.2419 - val_loss: 4.547
Epoch 15/20
446/446 ━━━━━━━━━━━━━━━━━━━━ 0s 522ms/step - accuracy: 0.6006 - loss: 0.9726
Epoch 15: ReduceLROnPlateau reducing learning rate to 2.499999936844688e-05.

Epoch 15: val_loss did not improve from 0.95141
446/446 ━━━━━━━━━━━━━━━━━━━━ 238s 534ms/step - accuracy: 0.6006 - loss: 0.9726 - val_accuracy: 0.6000 - val_loss: 0.951
Epoch 15: early stopping
Restoring model weights from the end of the best epoch: 5.
```

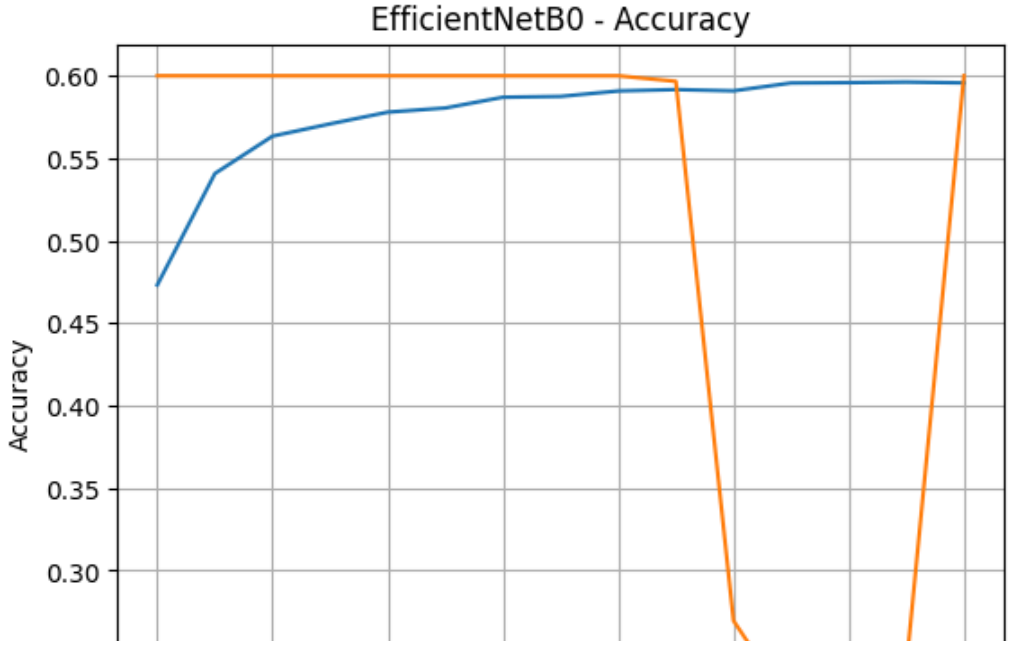
```
# === Function to plot training history ===
def plot_full_history(history, title):
    fig, ax = plt.subplots(1, 2, figsize=(14, 5))
    # Accuracy
    ax[0].plot(history.history['accuracy'], label='Train Accuracy')
    ax[0].plot(history.history['val_accuracy'], label='Validation Accuracy')
    ax[0].set_title(f'{title} - Accuracy')
    ax[0].set_xlabel('Epoch')
    ax[0].set_ylabel('Accuracy')
    ax[0].legend()
    ax[0].grid(True)
```

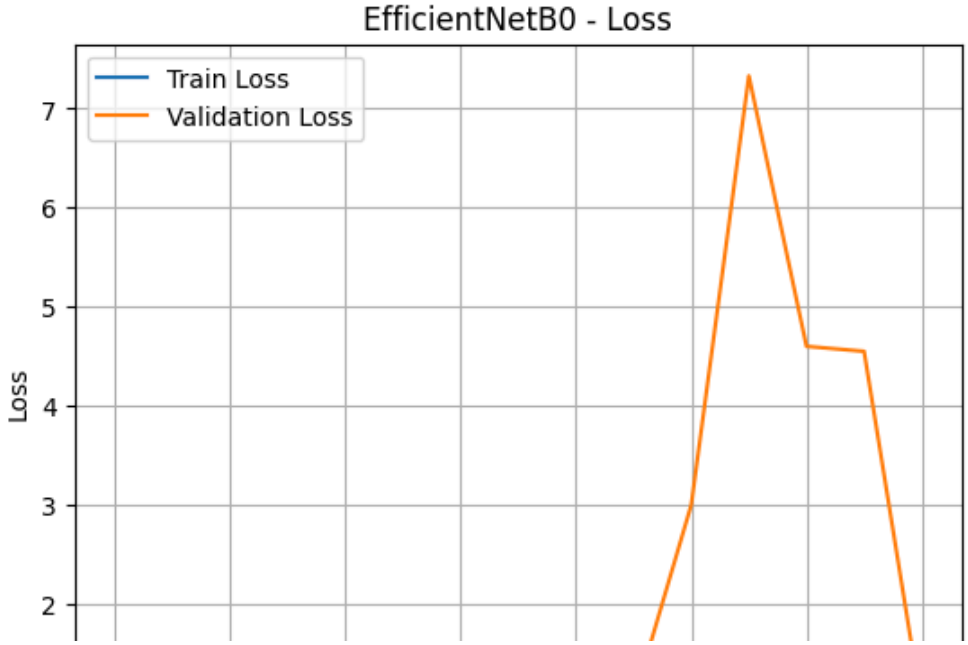
```
    # Loss
    ax[1].plot(history.history['loss'], label='Train Loss')
    ax[1].plot(history.history['val_loss'], label='Validation Loss')
    ax[1].set_title(f'{title} - Loss')
    ax[1].set_xlabel('Epoch')
    ax[1].set_ylabel('Loss')
    ax[1].legend()
    ax[1].grid(True)
    plt.show()
```

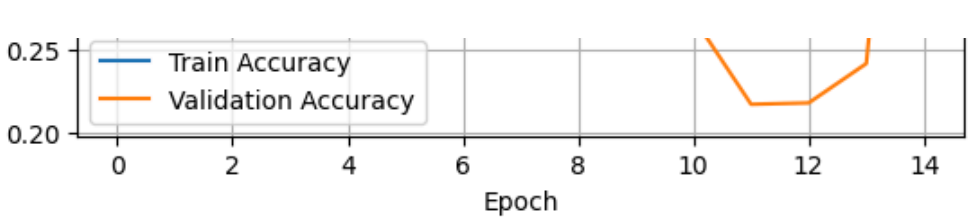
```
# === Plot training history for both models ===
plot_full_history(resnet_history, 'ResNet50')
plot_full_history(efficientnet_history, 'EfficientNetB0')
```

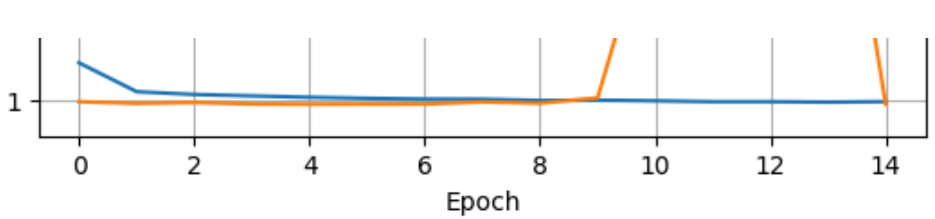

ResNet50 - Accuracy
Accuracy
Epoch
Train Accuracy
Validation Accuracy


ResNet50 - Loss
Loss
Epoch
Train Loss
Validation Loss


EfficientNetB0 - Accuracy
Accuracy


EfficientNetB0 - Loss
Loss
Train Loss
Validation Loss

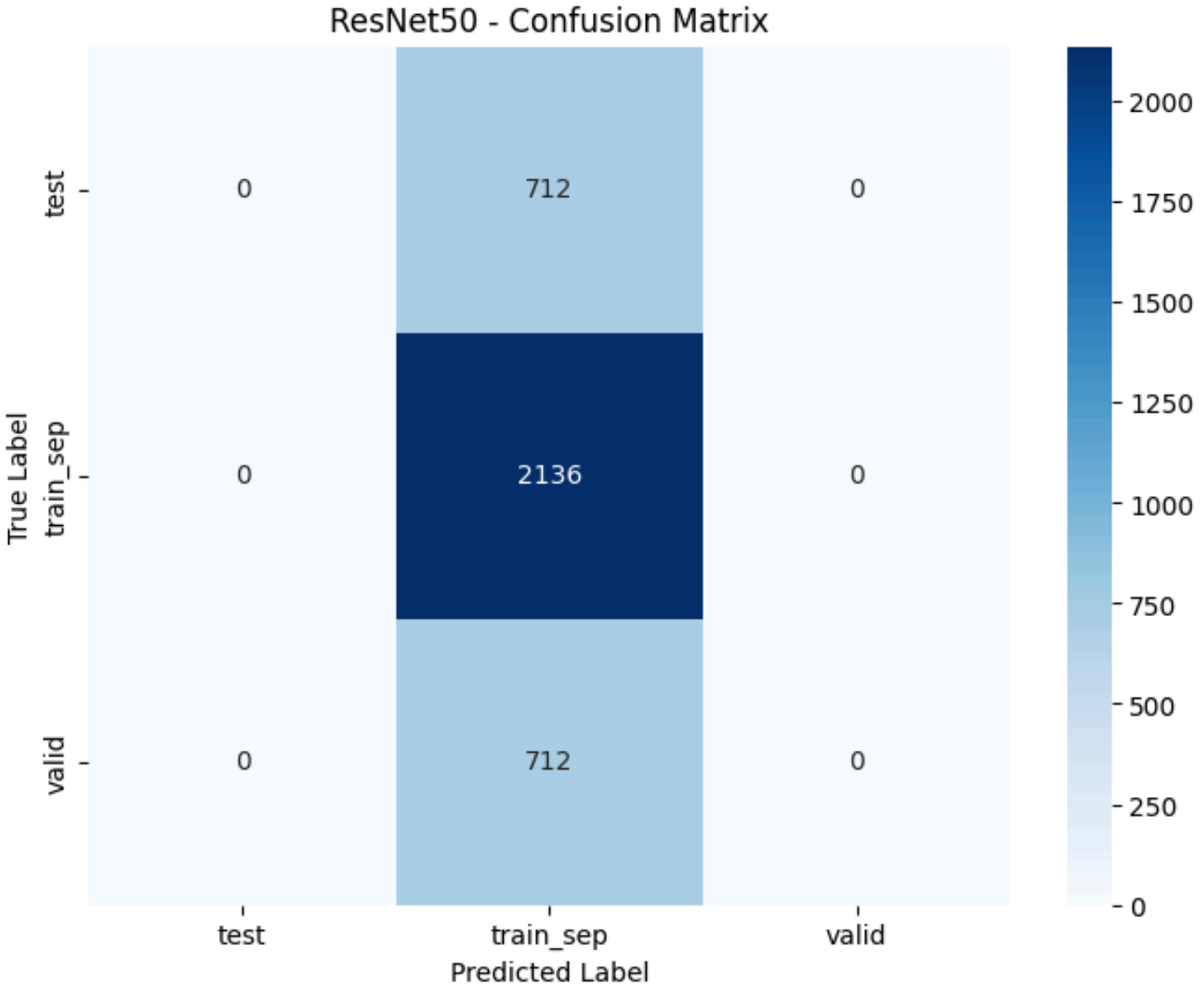





```
# === Evaluate Models ===
def evaluate_model(model, data, name):
    val_loss, val_acc = model.evaluate(data)
    print(f"\n{name} Validation Accuracy: {val_acc * 100:.2f}%")
    y_true = data.classes
    y_pred = np.argmax(model.predict(data), axis=1)
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=data.class_indices, yticklabels=data.class_indices)
    plt.title(f'{name} - Confusion Matrix')
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
    plt.show()
    print("\nClassification Report:\n")
    print(classification_report(y_true, y_pred, target_names=list(data.class_indices.keys())))
```

```
# === Evaluate both models ===
evaluate_model(resnet_model, val_data, "ResNet50")
evaluate_model(efficientnet_model, val_data, "EfficientNetB0")
```

```
112/112 ━━━━━━━━━━━━━━━━━━━━ 10s 85ms/step - accuracy: 0.5930 - loss: 1.0515

ResNet50 Validation Accuracy: 60.00%
112/112 ━━━━━━━━━━━━━━━━━━━━ 17s 117ms/step
```




```
Classification Report:
```