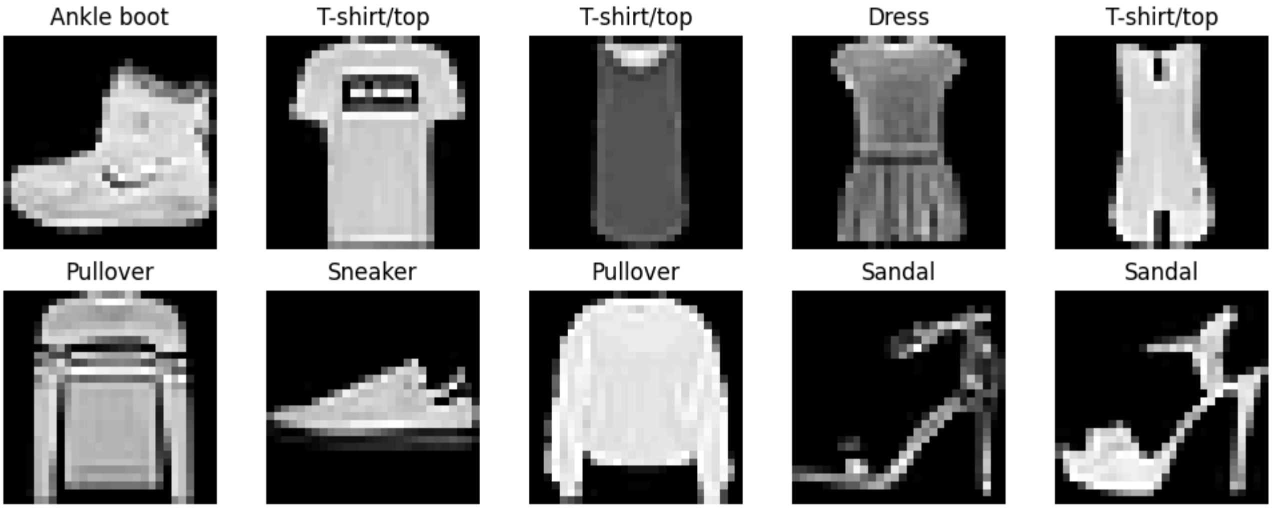
```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix

# Load Fashion MNIST
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

# Reduce size for Colab (optimized even more)
x_train, y_train = x_train[:3000], y_train[:3000]
x_test, y_test = x_test[:500], y_test[:500]

# Show 10 sample images
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
plt.figure(figsize=(10,4))
for i in range(10):
    plt.subplot(2, 5, i+1)
    plt.imshow(x_train[i], cmap='gray')
    plt.title(class_names[y_train[i]])
    plt.axis('off')
plt.tight_layout()
plt.show()

# Resize to (128x128) to save memory and convert grayscale to RGB
def preprocess_images(images):
    images = tf.expand_dims(images, -1)  # Add channel dim
    images = tf.image.resize(images, [128, 128])  # Reduce from 224 to 128
    images = tf.image.grayscale_to_rgb(images)
    images = images / 255.0  # Normalize
    return images.numpy()

x_train = preprocess_images(x_train)
x_test = preprocess_images(x_test)

# Validation split
x_val, y_val = x_train[-300:], y_train[-300:]
x_train, y_train = x_train[:-300], y_train[:-300]

# One-hot encode labels
y_train = to_categorical(y_train, 10)
y_val = to_categorical(y_val, 10)
y_test = to_categorical(y_test, 10)

# Data Augmentation (lighter)
datagen = ImageDataGenerator(rotation_range=10, zoom_range=0.05, horizontal_flip=True)
datagen.fit(x_train)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 ━━━━━━━━━━━━━━━━━━━━ 0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

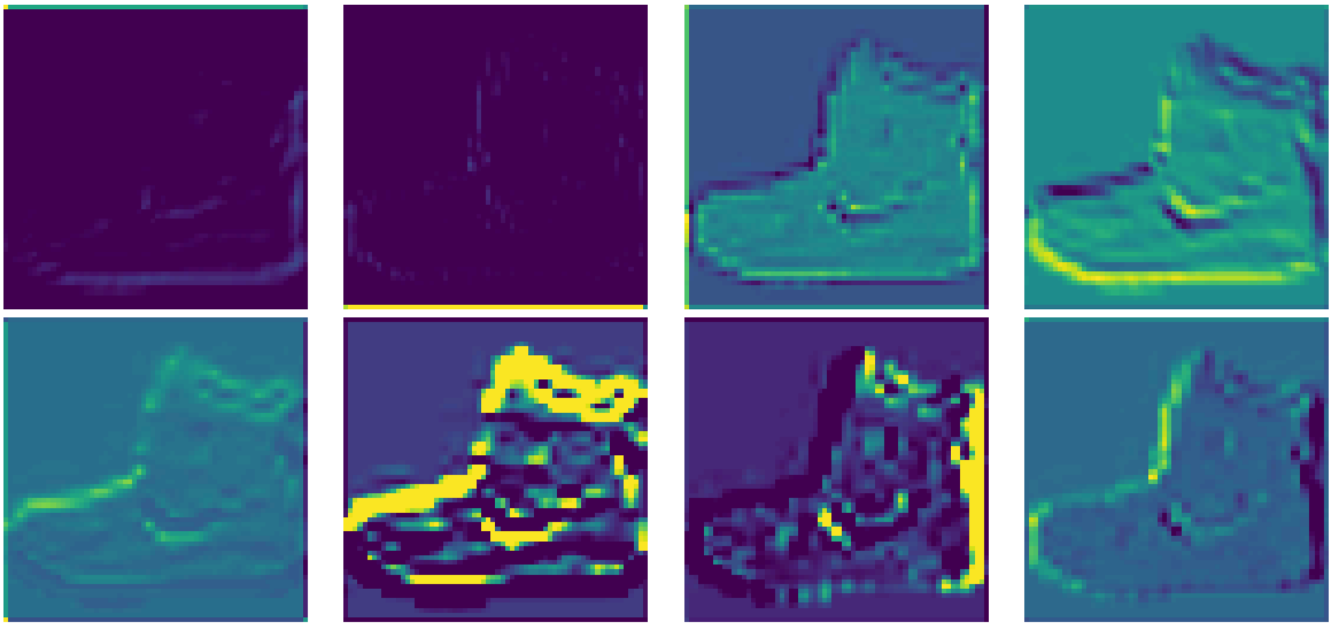



```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns

# Load dataset
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

# Reduce size for Colab
x_train, y_train = x_train[:5000], y_train[:5000]
x_test, y_test = x_test[:1000], y_test[:1000]

# Resize and convert to RGB
x_train = tf.image.resize_with_pad(tf.expand_dims(x_train, -1), 128, 128)
x_train = tf.image.grayscale_to_rgb(x_train)
x_test = tf.image.resize_with_pad(tf.expand_dims(x_test, -1), 128, 128)
x_test = tf.image.grayscale_to_rgb(x_test)

# Normalize
x_train, x_test = x_train / 255.0, x_test / 255.0
x_train, x_test = x_train.numpy(), x_test.numpy()

# Validation split
x_val, y_val = x_train[-500:], y_train[-500:]
x_train, y_train = x_train[:-500], y_train[:-500]

# One-hot encode
y_train = to_categorical(y_train, 10)
y_val = to_categorical(y_val, 10)
y_test = to_categorical(y_test, 10)

# Data Augmentation
datagen = ImageDataGenerator(rotation_range=15, zoom_range=0.1, horizontal_flip=True)
datagen.fit(x_train)

# Load MobileNetV2
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam

base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(128, 128, 3))

# Freeze base layers
for layer in base_model.layers:
    layer.trainable = False

# Add custom head
```

```
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.5)(x)
x = Dense(256, activation='relu')(x)
predictions = Dense(10, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

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

# Train model
history = model.fit(datagen.flow(x_train, y_train, batch_size=32),
                    validation_data=(x_val, y_val),
                    epochs=10)

# --------------------------
# Feature Map Visualization
# --------------------------
from tensorflow.keras.models import Model

# Choose a conv layer
layer_name = 'block_1_expand_relu'  # You can change this layer name
feature_model = Model(inputs=model.input, outputs=model.get_layer(layer_name).output)

# Get feature maps for one image
sample_image = x_train[0:1]
feature_maps = feature_model.predict(sample_image)

# Plot first 8 feature maps
plt.figure(figsize=(12, 6))
for i in range(8):
    plt.subplot(2, 4, i + 1)
    plt.imshow(feature_maps[0, :, :, i], cmap='viridis')
    plt.axis('off')
plt.suptitle(f'Feature Maps from layer: {layer_name}')
plt.tight_layout()
plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` cl
  self._warn_if_super_not_called()
Epoch 1/10
141/141 ━━━━━━━━━━━━━━━━━━━━ 39s 199ms/step - accuracy: 0.3187 - loss: 2.0435 - val_accuracy: 0.7960 - val_loss: 0.6196
Epoch 2/10
141/141 ━━━━━━━━━━━━━━━━━━━━ 18s 128ms/step - accuracy: 0.7137 - loss: 0.8882 - val_accuracy: 0.8520 - val_loss: 0.4400
Epoch 3/10
141/141 ━━━━━━━━━━━━━━━━━━━━ 17s 121ms/step - accuracy: 0.7464 - loss: 0.7284 - val_accuracy: 0.8540 - val_loss: 0.3849
Epoch 4/10
141/141 ━━━━━━━━━━━━━━━━━━━━ 18s 128ms/step - accuracy: 0.7545 - loss: 0.6619 - val_accuracy: 0.8480 - val_loss: 0.3832
Epoch 5/10
141/141 ━━━━━━━━━━━━━━━━━━━━ 17s 123ms/step - accuracy: 0.7707 - loss: 0.6292 - val_accuracy: 0.8780 - val_loss: 0.3580
Epoch 6/10
141/141 ━━━━━━━━━━━━━━━━━━━━ 18s 128ms/step - accuracy: 0.8009 - loss: 0.5509 - val_accuracy: 0.8800 - val_loss: 0.3339
Epoch 7/10
141/141 ━━━━━━━━━━━━━━━━━━━━ 19s 120ms/step - accuracy: 0.8079 - loss: 0.5455 - val_accuracy: 0.8820 - val_loss: 0.3219
Epoch 8/10
141/141 ━━━━━━━━━━━━━━━━━━━━ 21s 127ms/step - accuracy: 0.8209 - loss: 0.5019 - val_accuracy: 0.8940 - val_loss: 0.3090
Epoch 9/10
141/141 ━━━━━━━━━━━━━━━━━━━━ 17s 120ms/step - accuracy: 0.8131 - loss: 0.4945 - val_accuracy: 0.8880 - val_loss: 0.3085
Epoch 10/10
141/141 ━━━━━━━━━━━━━━━━━━━━ 18s 125ms/step - accuracy: 0.8212 - loss: 0.5035 - val_accuracy: 0.8920 - val_loss: 0.3041
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 525ms/step
```

Feature Maps from layer: block_1_expand_relu



```
# =====EVALUATE ON TEST DATA =======
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=1)
print(f"\nTest Accuracy: {test_acc:.4f}")
print(f"Test Loss: {test_loss:.4f}")

# ==PREDICTIONS ==
y_pred_probs = model.predict(x_test)
y_pred = np.argmax(y_pred_probs, axis=1)
y_true = np.argmax(y_test, axis=1)

# == CLASSIFICATION REPORT ==
print("\nClassification Report:\n")
print(classification_report(y_true, y_pred, target_names=[
    "T-shirt/top", "Trouser", "Pullover", "Dress", "Coat",
    "Sandal", "Shirt", "Sneaker", "Bag", "Ankle boot"
]))

# == CONFUSION MATRIX ==
conf_matrix = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
            xticklabels=[
                "T-shirt", "Trouser", "Pullover", "Dress", "Coat",
                "Sandal", "Shirt", "Sneaker", "Bag", "Boot"
            ],
            yticklabels=[
```

```
            yticklabels=[
                "T-shirt", "Trouser", "Pullover", "Dress", "Coat",
                "Sandal", "Shirt", "Sneaker", "Bag", "Boot"
            ])
plt.title("Confusion Matrix")
plt.ylabel("Actual")
plt.xlabel("Predicted")
plt.show()

# == ACCURACY & LOSS PLOTS ==
plt.figure(figsize=(14, 5))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Loss plot
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 270ms/step - accuracy: 0.8560 - loss: 0.3812

Test Accuracy: 0.8620
Test Loss: 0.3882
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 111ms/step

Classification Report:

              precision    recall  f1-score   support

 T-shirt/top       0.81      0.85      0.83       107
     Trouser       1.00      0.98      0.99       105
    Pullover       0.76      0.86      0.81       111
       Dress       0.88      0.84      0.86        93
        Coat       0.82      0.81      0.81       115
      Sandal       0.97      0.86      0.91        87
       Shirt       0.66      0.61      0.63        97
     Sneaker       0.86      0.91      0.88        95
         Bag       0.99      0.96      0.97        95
  Ankle boot       0.92      0.96      0.94        95

    accuracy                           0.86      1000
   macro avg       0.87      0.86      0.86      1000
weighted avg       0.86      0.86      0.86      1000
```