```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
import argparse

# Manually define args with default values
class Args:
    lr = 0.001
    batch_size = 64  # Increased batch size
    init = 2  # He initialization
    save_dir = "./saved_model/"

args = Args()


train_df = pd.read_csv("/content/fashion-mnist_train.csv")
test_df = pd.read_csv("/content/fashion-mnist_test.csv")

X_train = train_df.drop('label', axis=1).values.reshape(-1, 28, 28, 1).astype('float32') / 255.0
y_train = train_df['label'].values
X_test = test_df.drop('label', axis=1).values.reshape(-1, 28, 28, 1).astype('float32') / 255.0
y_test = test_df['label'].values



datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
)

datagen.fit(X_train)



model = models.Sequential([
    layers.Conv2D(64, (3, 3), activation='relu', padding='same', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(256, (3, 3), activation='relu', padding='same'),
    layers.Conv2D(256, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(1024, activation='relu'),
    layers.Dropout(0.5),  # Added dropout layer
    layers.Dense(1024, activation='relu'),
    layers.Dropout(0.5),  # Added dropout layer
    layers.Dense(10, activation='softmax')
])



model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=args.lr),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
history = model.fit(datagen.flow(X_train, y_train, batch_size=args.batch_size),
                    steps_per_epoch=len(X_train) / args.batch_size, epochs=100,
                    validation_data=(X_test, y_test), callbacks=[early_stopping])
```

```
Epoch 1/100
937/937 [==============================] - 704s 745ms/step - loss: 0.6491 - accuracy: 0.7520 - val_loss: 0.3905 - val_accuracy: 0.8505
Epoch 2/100
937/937 [==============================] - 698s 744ms/step - loss: 0.4145 - accuracy: 0.8464 - val_loss: 0.3305 - val_accuracy: 0.8769
Epoch 3/100
937/937 [==============================] - 704s 751ms/step - loss: 0.3670 - accuracy: 0.8641 - val_loss: 0.2650 - val_accuracy: 0.9013
Epoch 4/100
937/937 [==============================] - 704s 752ms/step - loss: 0.3405 - accuracy: 0.8759 - val_loss: 0.2741 - val_accuracy: 0.8980
Epoch 5/100
937/937 [==============================] - 696s 743ms/step - loss: 0.3196 - accuracy: 0.8834 - val_loss: 0.2664 - val_accuracy: 0.8995
Epoch 6/100
937/937 [==============================] - 697s 743ms/step - loss: 0.3121 - accuracy: 0.8841 - val_loss: 0.2648 - val_accuracy: 0.8980
```

```
Epoch 7/100
937/937 [==============================] - 698s 745ms/step - loss: 0.2980 - accuracy: 0.8903 - val_loss: 0.2410 - val_accuracy: 0.9105
Epoch 8/100
937/937 [==============================] - 685s 731ms/step - loss: 0.2917 - accuracy: 0.8924 - val_loss: 0.2364 - val_accuracy: 0.9121
Epoch 9/100
937/937 [==============================] - 703s 750ms/step - loss: 0.2841 - accuracy: 0.8948 - val_loss: 0.2489 - val_accuracy: 0.9049
Epoch 10/100
937/937 [==============================] - 697s 744ms/step - loss: 0.2784 - accuracy: 0.8963 - val_loss: 0.2203 - val_accuracy: 0.9143
Epoch 11/100
937/937 [==============================] - 703s 750ms/step - loss: 0.2746 - accuracy: 0.8997 - val_loss: 0.2175 - val_accuracy: 0.9192
Epoch 12/100
937/937 [==============================] - 698s 744ms/step - loss: 0.2732 - accuracy: 0.9017 - val_loss: 0.2309 - val_accuracy: 0.9120
Epoch 13/100
937/937 [==============================] - 703s 750ms/step - loss: 0.2693 - accuracy: 0.9012 - val_loss: 0.2672 - val_accuracy: 0.9010
Epoch 14/100
937/937 [==============================] - 703s 750ms/step - loss: 0.2667 - accuracy: 0.9021 - val_loss: 0.2357 - val_accuracy: 0.9169
Epoch 15/100
937/937 [==============================] - 685s 731ms/step - loss: 0.2611 - accuracy: 0.9054 - val_loss: 0.2305 - val_accuracy: 0.9121
Epoch 16/100
937/937 [==============================] - 695s 742ms/step - loss: 0.2579 - accuracy: 0.9060 - val_loss: 0.2231 - val_accuracy: 0.9195
```

```
model.save(args.save_dir + 'fashion_mnist_cnn_model.h5')
```

```
test_loss, test_acc = model.evaluate(X_test, y_test)
print("Test Accuracy:", test_acc)
```

```
313/313 [==============================] - 35s 110ms/step - loss: 0.2175 - accuracy: 0.9192
Test Accuracy: 0.9192000031471252
```

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
import argparse

# Manually define args with default values
class Args:
    lr = 0.001
    batch_size = 64  # Increased batch size
    init = 2  # He initialization
    save_dir = "./saved_model/"

args = Args()
train_df = pd.read_csv("/content/fashion-mnist_train.csv")
test_df = pd.read_csv("/content/fashion-mnist_test.csv")

X_train = train_df.drop('label', axis=1).values.reshape(-1, 28, 28, 1).astype('float32') / 255.0
y_train = train_df['label'].values
X_test = test_df.drop('label', axis=1).values.reshape(-1, 28, 28, 1).astype('float32') / 255.0
y_test = test_df['label'].values
datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
)

datagen.fit(X_train)
model = models.Sequential([
    layers.Conv2D(64, (3, 3), activation='relu', padding='same', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(256, (3, 3), activation='relu', padding='same'),
    layers.Conv2D(256, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(1024, activation='relu'),
    layers.Dropout(0.5),  # Added dropout layer
    layers.Dense(1024, activation='relu'),
    layers.Dropout(0.5),  # Added dropout layer
    layers.Dense(10, activation='softmax')
])
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=args.lr),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
history = model.fit(datagen.flow(X_train, y_train, batch_size=args.batch_size),
                    steps_per_epoch=len(X_train) / args.batch_size, epochs=100,
                    validation_data=(X_test, y_test), callbacks=[early_stopping])
model.save(args.save_dir + 'fashion_mnist_cnn_model.h5')
test_loss, test_acc = model.evaluate(X_test, y_test)
print("Test Accuracy:", test_acc)
```

```
Epoch 1/100
598/598 [==============================] - 417s 694ms/step - loss: nan - accuracy: 0.1298 - val_loss: nan - val_accuracy: 0.1000
Epoch 2/100
598/598 [==============================] - 414s 691ms/step - loss: nan - accuracy: 0.0994 - val_loss: nan - val_accuracy: 0.1000
Epoch 3/100
598/598 [==============================] - 425s 710ms/step - loss: nan - accuracy: 0.0994 - val_loss: nan - val_accuracy: 0.1000
Epoch 4/100
598/598 [==============================] - 420s 703ms/step - loss: nan - accuracy: 0.0994 - val_loss: nan - val_accuracy: 0.1000
Epoch 5/100
598/598 [==============================] - 403s 674ms/step - loss: nan - accuracy: 0.0994 - val_loss: nan - val_accuracy: 0.1000
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `m
  saving_api.save_model(
313/313 [==============================] - 25s 81ms/step - loss: nan - accuracy: 0.1000
Test Accuracy: 0.10000000149011612
```

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.model_selection import train_test_split

# Manually define args with default values
class Args:
    lr = 0.001
    batch_size = 128  # Increased batch size
    init = 2  # He initialization
    save_dir = "./saved_model/"

args = Args()

# Load dataset
train_df = pd.read_csv("/content/fashion-mnist_train.csv")
test_df = pd.read_csv("/content/fashion-mnist_test.csv")

# Prepare data
X_train = train_df.drop('label', axis=1).values.reshape(-1, 28, 28, 1).astype('float32') / 255.0
y_train = train_df['label'].values
X_test = test_df.drop('label', axis=1).values.reshape(-1, 28, 28, 1).astype('float32') / 255.0
y_test = test_df['label'].values

# Data augmentation
datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
)
datagen.fit(X_train)

# Define the model
model = models.Sequential([
    layers.Conv2D(64, (3, 3), activation='relu', padding='same', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(256, (3, 3), activation='relu', padding='same'),
    layers.Conv2D(256, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(1024, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1024, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(10, activation='softmax')
])

# Compile the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=args.lr),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
model_checkpoint = ModelCheckpoint(args.save_dir + 'best_model.h5', monitor='val_accuracy', save_best_only=True, mode='max', verbose=1)

# Train the model
history = model.fit(datagen.flow(X_train, y_train, batch_size=args.batch_size),
                    steps_per_epoch=len(X_train) / args.batch_size, epochs=100,
                    validation_data=(X_test, y_test), callbacks=[early_stopping, model_checkpoint])

# Save the model
model.save(args.save_dir + 'fashion_mnist_cnn_model.h5')

# Evaluate the model
test_loss, test_acc = model.evaluate(X_test, y_test)
print("Test Accuracy:", test_acc)
```

```
Epoch 1/100
469/468 [==============================] - ETA: 0s - loss: 0.6898 - accuracy: 0.7358
Epoch 1: val_accuracy improved from -inf to 0.83890, saving model to ./saved_model/best_model.h5
468/468 [==============================] - 556s 1s/step - loss: 0.6898 - accuracy: 0.7358 - val_loss: 0.4240 - val_accuracy: 0.8389
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via
  saving_api.save_model(
Epoch 2/100
469/468 [==============================] - ETA: 0s - loss: 0.4276 - accuracy: 0.8403
Epoch 2: val_accuracy improved from 0.83890 to 0.87310, saving model to ./saved_model/best_model.h5
468/468 [==============================] - 567s 1s/step - loss: 0.4276 - accuracy: 0.8403 - val_loss: 0.3373 - val_accuracy: 0.8731
Epoch 3/100
469/468 [==============================] - ETA: 0s - loss: 0.3704 - accuracy: 0.8607
Epoch 3: val_accuracy improved from 0.87310 to 0.88940, saving model to ./saved_model/best_model.h5
468/468 [==============================] - 570s 1s/step - loss: 0.3704 - accuracy: 0.8607 - val_loss: 0.2984 - val_accuracy: 0.8894
Epoch 4/100
469/468 [==============================] - ETA: 0s - loss: 0.3354 - accuracy: 0.8747
Epoch 4: val_accuracy did not improve from 0.88940
468/468 [==============================] - 564s 1s/step - loss: 0.3354 - accuracy: 0.8747 - val_loss: 0.2984 - val_accuracy: 0.8838
Epoch 5/100
469/468 [==============================] - ETA: 0s - loss: 0.3173 - accuracy: 0.8837
Epoch 5: val_accuracy improved from 0.88940 to 0.90470, saving model to ./saved_model/best_model.h5
468/468 [==============================] - 544s 1s/step - loss: 0.3173 - accuracy: 0.8837 - val_loss: 0.2551 - val_accuracy: 0.9047
Epoch 6/100
469/468 [==============================] - ETA: 0s - loss: 0.2989 - accuracy: 0.8892
Epoch 6: val_accuracy improved from 0.90470 to 0.90900, saving model to ./saved_model/best_model.h5
468/468 [==============================] - 566s 1s/step - loss: 0.2989 - accuracy: 0.8892 - val_loss: 0.2433 - val_accuracy: 0.9090
Epoch 7/100
```