Задача 1

За решаване на задачата локално на нашия компютър се изисква да се инсталира [python 3](https://www.python.org/downloads/) и [tensorflow](https://pypi.org/project/tensorflow/) (чрез командата **pip install tensorflow**). Вместо това реших да използвам [Google Colab](https://colab.research.google.com/), тъй като е доста удобна услуга, която избягва повечето настройки/сваляне на пакети (ползва се тяхно вече настроено устройство) и отделно ни предоставя безплатен достъп до повече ресурси (удобно за хора със стари и слаби устройства).

1. Изпълнение на кода без правене на допълнителни промени

**КОД 1 (без Callback):**

import tensorflow as tf

data = tf.keras.datasets.fashion\_mnist

(training\_images, training\_labels), (test\_images, test\_labels) = data.load\_data()

training\_images = training\_images / 255.0

test\_images = test\_images / 255.0

model = tf.keras.models.Sequential([

tf.keras.layers.Flatten(input\_shape=(28, 28)),

tf.keras.layers.Dense(128, activation=tf.nn.relu),

tf.keras.layers.Dense(10, activation=tf.nn.softmax)

])

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

model.fit(training\_images, training\_labels, epochs=5)

model.evaluate(test\_images, test\_labels)

classifications = model.predict(test\_images)

print(classifications[0])

print(test\_labels[0])

**Резултат:**

Epoch 1/5

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 5ms/step - accuracy: 0.7787 - loss: 0.6355

Epoch 2/5

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.8640 - loss: 0.3811

Epoch 3/5

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8743 - loss: 0.3419

Epoch 4/5

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8835 - loss: 0.3162

Epoch 5/5

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 4ms/step - accuracy: 0.8920 - loss: 0.2969

313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8766 - loss: 0.3467

313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step

[2.0416655e-06 4.7499600e-07 5.5443256e-06 1.9083111e-05 3.0317403e-07

2.4081904e-03 2.2606785e-06 5.6264348e-02 5.5354580e-05 9.4124240e-01]

9

**Коментар:**

Епохи: 5

Точност след 5 епохи:

Тренировъчна точност: 89.20%

Точност на теста: 87.66%

Загуба след 5 епохи:

Тренировъчна загуба: 0.2969

Загуба на теста: 0.3467

Най-висока вероятност: **0.941** (изображението принадлежи на **9 та класа**)

**КОД 2 (с Callback):**

import tensorflow as tf

data = tf.keras.datasets.fashion\_mnist

class myCallback(tf.keras.callbacks.Callback):

def on\_epoch\_end(self, epoch, logs={}):

if(logs.get('accuracy')>0.85):

print("\nReached 85% accuracy so cancelling training!")

self.model.stop\_training = True

callbacks = myCallback()

(training\_images, training\_labels), (test\_images, test\_labels) = data.load\_data()

training\_images=training\_images.reshape(60000, 28, 28, 1)

training\_images = training\_images / 255.0

test\_images = test\_images.reshape(10000, 28, 28, 1)

test\_images = test\_images / 255.0

model = tf.keras.models.Sequential([

tf.keras.layers.Flatten(input\_shape=(28, 28)),

tf.keras.layers.Dense(128, activation=tf.nn.relu),

tf.keras.layers.Dense(10, activation=tf.nn.softmax)

])

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

model.fit(training\_images, training\_labels, epochs=5, callbacks=[callbacks])

model.evaluate(test\_images, test\_labels)

classifications = model.predict(test\_images)

print(classifications[0])

print(test\_labels[0])

**Резултат:**

Epoch 1/5

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.7821 - loss: 0.6212

Epoch 2/5

1872/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8642 - loss: 0.3823

Reached 85% accuracy so cancelling training!

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8642 - loss: 0.3823

313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8635 - loss: 0.3805

313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step

[1.9433117e-05 5.1375963e-07 1.6387289e-06 8.2938072e-07 2.4775948e-06

2.4871875e-02 6.1826290e-06 2.3240338e-01 2.9366747e-03 7.3975682e-01]

9

**Коментар:**

Епохи: 5

Точност след 1 епоха: 86.42% (Тренировка)

Callback: Обучението беше спряно рано, след като достигна 85% точност в първата епоха благодарение на персонализирания Callback.

Точност на теста: 86.35%

Загуба след 1 епоха:

Тренировъчна загуба: 0.3823

Загуба на теста: 0.3805

Прогнози: Подобно на първото изпълнение, моделът прогнозира вероятностите за всяка класа и отново, най-високата вероятност е за **9-та класа** с вероятност **0.739**.

**Извод:**

* Първото изпълнение завършва всички 5 епохи и постига по-добра точност (89.20% на тренировка и 87.66% на теста).
* Второто изпълнение използва персонализиран callback, който спира обучението след достигане на 85% точност в първата епоха. Това води до по-кратко време на обучение, но с малко по-ниска точност (86.42% на тренировка и 86.35% на теста).
* Точността на теста и в двата случая е доста добра, около 86-87%, което показва, че моделът постига добри резултати на набора от данни Fashion MNIST.

1. Различни начини за подобрение на кода

**КОД 1 (без Callback)**:

* увеличаване на броя на епохите
* експериментиране с други архитектурни промени (например добавяне на повече слоеве или настройване на темпото на учене).

**КОД 2 (с Callback)**:

* Callback функцията може да бъде полезна за предотвратяване на преобучение или за намаляване на времето за обучение. Въпреки това, ако целта е по-висока точност, може да се наложи да тренираме повече епохи или да коригираме условието на callback-а (например, да спрем при 90% точност).

1. Подобряване на кода и тестване

**КОД 1 (без Callback)**:

import tensorflow as tf

data = tf.keras.datasets.fashion\_mnist

(training\_images, training\_labels), (test\_images, test\_labels) = data.load\_data()

# Рескалиране на изображенията и подготвяне за моделиране

training\_images = training\_images.reshape(60000, 28, 28, 1) / 255.0

test\_images = test\_images.reshape(10000, 28, 28, 1) / 255.0

# Създаване на по-сложен модел с допълнителни слоеве

model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),

tf.keras.layers.MaxPooling2D((2, 2)),

tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),

tf.keras.layers.MaxPooling2D((2, 2)),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(128, activation='relu'),

tf.keras.layers.Dense(10, activation='softmax')

])

# Компилация на модела

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

# Този път увеличаваме епохите да са 10, вместо 5

model.fit(training\_images, training\_labels, epochs=10)

# Оценка на модела на тестови данни

test\_loss, test\_accuracy = model.evaluate(test\_images, test\_labels)

print(f"Точност на теста: {test\_accuracy\*100:.2f}%")

# Прогноза

classifications = model.predict(test\_images)

print(classifications[0])

print(test\_labels[0])

**Направени промени:**

* Увеличение на броя на епохите да е 10 вместо 5
* Замяна на плътните слоеве с конволюционни слоеве за по-добро представяне на модела.
* Добавяне на допълнителни слоеве за плоски (Flatten) и пулни (MaxPooling), които помагат на модела да извлече характеристики от изображенията по-ефективно.
* Правилна нормализация на изображенията

**Резултат:**

Epoch 1/10

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 58s 30ms/step - accuracy: 0.7783 - loss: 0.6189

Epoch 2/10

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 83s 30ms/step - accuracy: 0.8860 - loss: 0.3097

Epoch 3/10

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 79s 29ms/step - accuracy: 0.9064 - loss: 0.2507

Epoch 4/10

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 84s 30ms/step - accuracy: 0.9151 - loss: 0.2244

Epoch 5/10

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 56s 30ms/step - accuracy: 0.9281 - loss: 0.1906

Epoch 6/10

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 60s 32ms/step - accuracy: 0.9381 - loss: 0.1651

Epoch 7/10

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 75s 28ms/step - accuracy: 0.9434 - loss: 0.1482

Epoch 8/10

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 83s 29ms/step - accuracy: 0.9522 - loss: 0.1284

Epoch 9/10

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 83s 29ms/step - accuracy: 0.9579 - loss: 0.1138

Epoch 10/10

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 80s 28ms/step - accuracy: 0.9629 - loss: 0.0964

313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - accuracy: 0.9126 - loss: 0.2897

Точност на теста: 91.59%

313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step

[6.6574034e-07 5.3573476e-11 2.5064921e-08 6.0167255e-10 1.2467592e-10

6.7921761e-08 1.5538846e-11 8.5145746e-07 8.4540880e-10 9.9999827e-01]

9

**Коментар:**

Епохи: 10 епохи общо

Крайна точност на обучението: 96.29%

Крайна точност на теста: 91.59%

Точността на теста показва колко добре моделът е работил с данни, които не е виждал досега.

В този случай, вероятността от **99.9998%** е за **9-ти клас** (индекс 9).

* Точността на обучението се подобрява постоянно през епохите, което показва, че моделът се учи и подобрява с времето.
* Точността на теста (91.59%) е малко по-ниска от точността на обучението (96.29%), което е нормално. Малката разлика между точността на обучението и теста предполага, че моделът е добър без да има overfitting.
* Точността на теста е висока, което показва, че моделът е ефективен в предсказанията за нови данни.

**Подобренията на кода бяха успешни.**

**КОД 2 (с Callback)**:

import tensorflow as tf

data = tf.keras.datasets.fashion\_mnist

# Създаване на Callback за ранно спиране при достигане на точност над 90% (този път искаме по голяма точност сравнение с оригиналното)

class MyCallback(tf.keras.callbacks.Callback):

def on\_epoch\_end(self, epoch, logs={}):

if logs.get('accuracy') > 0.90:

print("\n Достигнахме 90% точност, спираме обучението!")

self.model.stop\_training = True

callbacks = MyCallback()

(training\_images, training\_labels), (test\_images, test\_labels) = data.load\_data()

# Рескалиране на изображенията и подготвяне за моделиране

training\_images = training\_images.reshape(60000, 28, 28, 1) / 255.0

test\_images = test\_images.reshape(10000, 28, 28, 1) / 255.0

# Създаване на по-сложен модел с допълнителни слоеве

model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),

tf.keras.layers.MaxPooling2D((2, 2)),

tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),

tf.keras.layers.MaxPooling2D((2, 2)),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(128, activation='relu'),

tf.keras.layers.Dense(10, activation='softmax')

])

# Компилация на модела

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

# Обучение на модела с Callback за ранно спиране и увеличение на броя епохи

model.fit(training\_images, training\_labels, epochs=10, callbacks=[callbacks])

# Оценка на модела на тестови данни

test\_loss, test\_accuracy = model.evaluate(test\_images, test\_labels)

print(f"Точност на теста: {test\_accuracy\*100:.2f}%")

# Прогноза

classifications = model.predict(test\_images)

print(classifications[0])

print(test\_labels[0])

**Направени промени:**

* Увеличение на броя на епохите да е 10 вместо 5
* Модела ползва конволюционни слоеве, за да се подобри представянето.
* Добавяне на callback за ранно спиране, който спира обучението, когато точността надвиши 90% (искаме по-голяма точност сравнение с оригинала).
* Правилна нормализация на изображенията

**Резултат:**

Epoch 1/10

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 57s 30ms/step - accuracy: 0.7761 - loss: 0.6241

Epoch 2/10

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 55s 29ms/step - accuracy: 0.8866 - loss: 0.3104

Epoch 3/10

1874/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.9038 - loss: 0.2577

Достигнахме 90% точност, спираме обучението!

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 82s 29ms/step - accuracy: 0.9038 - loss: 0.2576

313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - accuracy: 0.8985 - loss: 0.2757

Точност на теста: 90.12%

313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step

[1.6251851e-06 9.8438875e-07 1.5655090e-06 4.0082817e-09 5.1076594e-08

3.8692809e-04 5.2738091e-07 4.6498217e-03 6.0604456e-05 9.9489796e-01]

9

**Коментар:**

Епохи: 10 епохи общо

Крайна точност на обучението: 90.38% (преди спирането на обучението поради callback)

Крайна точност на теста: 90.12%

Вероятността от **99.48%** е за **9-ти клас** (индекс 9).

* Точността на обучението се подобри значително през първите 2 епохи и спря да се увеличава след 3-та епоха. Това се случи, защото обучението беше прекратено, когато моделът достигна 90% точност, съгласно зададения callback.
* Точността на теста е много близка до точността на обучението, като има разлика от само 0.26%. Това показва, че моделът е добър без да има overfitting.
* Точността на теста е добра, показвайки, че моделът дава стабилни и надеждни прогнози.

**Подобренията на кода бяха успешни.**

**Самия проект е качен** [**тук**](https://github.com/nikoladevelops/machine-learning-exercise/tree/main)**:**