Logo

Description automatically generated

VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

**NERELIACINĖS DUOMENŲ BAZĖS**

Situacijos analizė

Rengė: Tomas Kvedaravičius

Informacinių sistemų inžinerija, III kursas, I grupė

Vadovas: Giedrius Graževičius

Vilnius, 2022

Turinys

1. [Užduoties tikslas](#_1._Užduoties_tikslas)
2. [Darbo eiga](#_3._Darbo_eiga)

2.1. [Duomenų paruošimas](#_3.1._Duomenų_paruošimas)

2.2 [Programos kūrimas](#_3.2._Programos_kūrimas)

2.3 [Tinklo architektūros parinkimas](#_3.3._Tinklo_architektūros)

2.4 [Tinklo parametrų parinkimas](#_3.4._Tinklo_parametrų)

1. [Naudoti šaltiniai](#_5._Naudoti_šaltiniai)
2. [Priedai](#_6._Priedai)

# 1. Užduoties tikslas

Apmokyti konvoliucinį neuroninį tinklą vaizdams klasifikuoti, atlikti tyrimą.

# 2. Anotacija

Konvoliucinio tinklo apmokymui ir tyrimui atlikti buvo naudojamas CIFAR 10 vaizdų rinkinys. Šiame rinkinyje yra 60000 duomenų: 50000 mokymo ir 10000 testavimo. Duomenys buvo įkelti *tensorflow* bibliotekos pagalbą.

# 3. Darbo eiga

## 3.1. Duomenų paruošimas

Duomenys buvo paimti iš *tenserflow* paketo su funkcija *cifar10.load\_data().* Ši funkcija gražina duomenys kurie yra padalinti į du rinkinius:

* *X\_train*, kuriame yra 50000 įrašų;
* *X\_test*, kuriame yra 10000 nepanaudotų įrašų;

Duomenys yra įkeliami kiekviena kartą pakvietus funkciją. Duomenų rinkinys turi 10 klasių:

„Airplane“, „Automobile“, „Bird“, „Cat“, „Deer“, „Dog“, „Frog“, „Horse“, „Ship“, „Truck“.

Naudojantis šia funkcija duomenys automatiškai yra suskirstomi į mokymosi ir testavimo aibes santykiu 80/20. Šis santykis atitinka gerąsias praktikas, todėl nebuvo modifikuojamas duomenų skaidymas.

## 3.2. Programos kūrimas

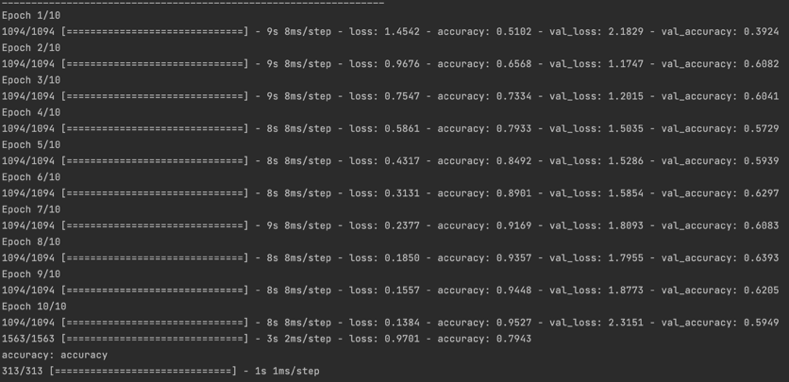
Remiantis keliais interneto šaltiniais buvo sukurta programa, kuri klasifikuoja vaizdus į atitinkamas klases. Programa naudoją procesoriaus skaičiavimo resursus, nes dėl techninių kliūčių nepavyko įdiegti grafinės kortos programinės įrangos. Programa buvo leidžiama ant ubuntu serverio (Ubuntu 22.04.1 LTS) *PyCharm* programinės įrangos pagalbą. Programa buvo sukurta remiantis 4 išvardytais šaltiniais ir pasinaudojant juose esančia informacija.

## 3.3. Tinklo architektūros parinkimas

Norinti atrasti geriausią tinklo architektūrą buvo išbandyti 4 tinklo modeliai. Iš jų buvo pasirinktas modelis, kuris kokybiškiausiai klasifikuoja paveikslėlius esant fiksuotiems parametrams.

Pirmasis modelis:

Text

Description automatically generated with medium confidence 

1 pav. Pirmojo modelio architektūra ir mokymasi rezultatai

A picture containing graphical user interface

Description automatically generated Chart, line chart

Description automatically generated

2 pav. Pirmojo modelio klasifikavimo matrica ir grafikas

Antrasis modelis:

A picture containing text

Description automatically generated Text

Description automatically generated

3 pav. Antrojo modelio architektūra ir mokymasi rezultatai

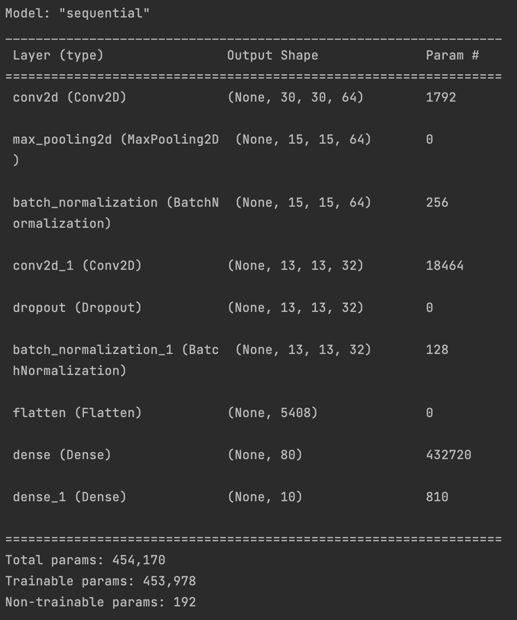
Chart

Description automatically generated with medium confidenceChart, line chart

Description automatically generated

4 pav. Antrojo modelio klasifikavimo matrica ir grafikas

Trečiasis modelis:

 Text

Description automatically generated

5 pav. Trečiojo modelio architektūra ir mokymasi rezultatai

A picture containing graphical user interface

Description automatically generated Chart, line chart

Description automatically generated

6 pav. Trečiojo modelio klasifikavimo matrica ir grafikas

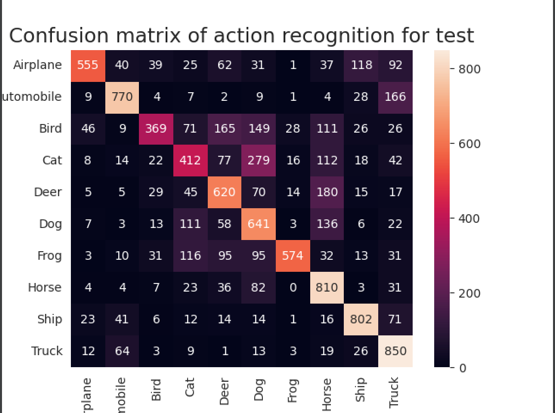
Ketvirtasis modelis:

Text

Description automatically generated with medium confidence Text

Description automatically generated

7 pav. Ketvirtojo modelio architektūra ir mokymasi rezultatai

Chart, line chart

Description automatically generated

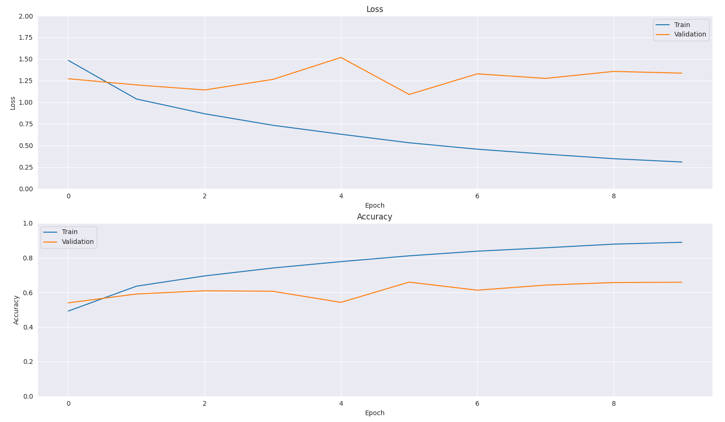
8 pav. Ketvirtojo modelio klasifikavimo matrica ir grafikas

Esant vienodiems parametrams trečiasis modelis buvo tiksliausias, todėl jis bus naudojamas tinklo parametrų parinkimui. Iš architektūrų ekranų paveikslų galima matyti, kad modelis su vienu *Dropout* buvo tiksliausias. Jo rezultatai nebuvo 100 %, bet tai bandysime pasiekti keičiant tinklo parametrus.

## 3.4. Tinklo parametrų parinkimas

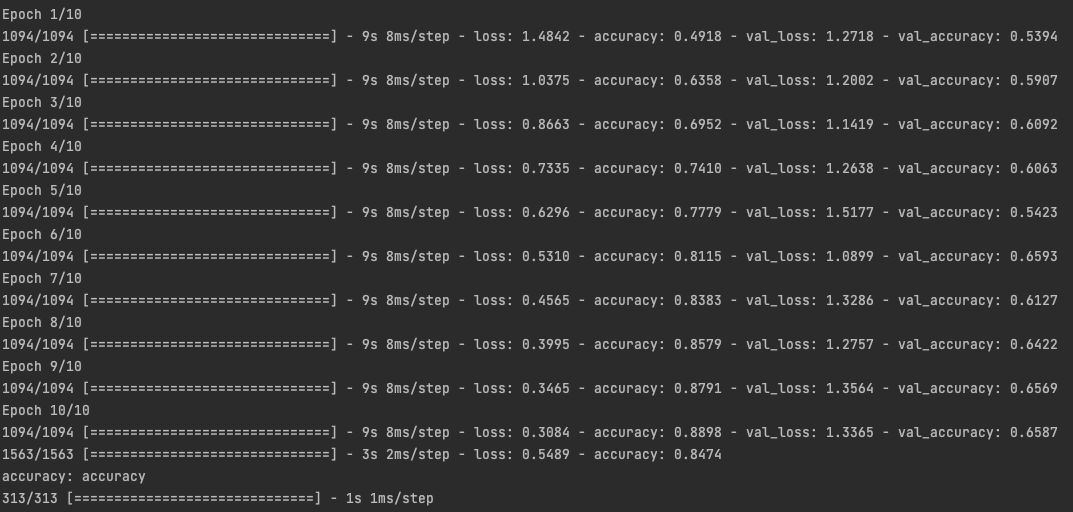
Pirmam bandymui buvo pasirinkti tokie parametrai:

* *validation\_split* – 0,3
* *epochs* – 10
* *learningRate* – 0,001

A picture containing chart

Description automatically generated

9 pav. Pirmojo bandymo tiklsumo ir nuostolio grafikas ir matrica



10 pav. Pirmojo bandymo mokymosi rezultatai

Klasifikavimo tikslumas testavimo duomenims gautas neblogas 84,74 %.

Antro bandymo parametrai:

* *validation\_split* – 0,5
* *epochs* – 10
* *learningRate* – 0,01

Chart, line chart

Description automatically generatedGraphical user interface

Description automatically generated with low confidence

11 pav. Antrojo bandymo tiklsumo ir nuostolio grafikas ir matrica

Text

Description automatically generated

12 pav. Antrojo bandymo mokymosi rezultatai

Klasifikavimo tikslumas testavimo duomenims gautas 56,46 %. Tai yra stipriai mažiau pirmojo bandymo metu.

Trečio bandymo parametrai:

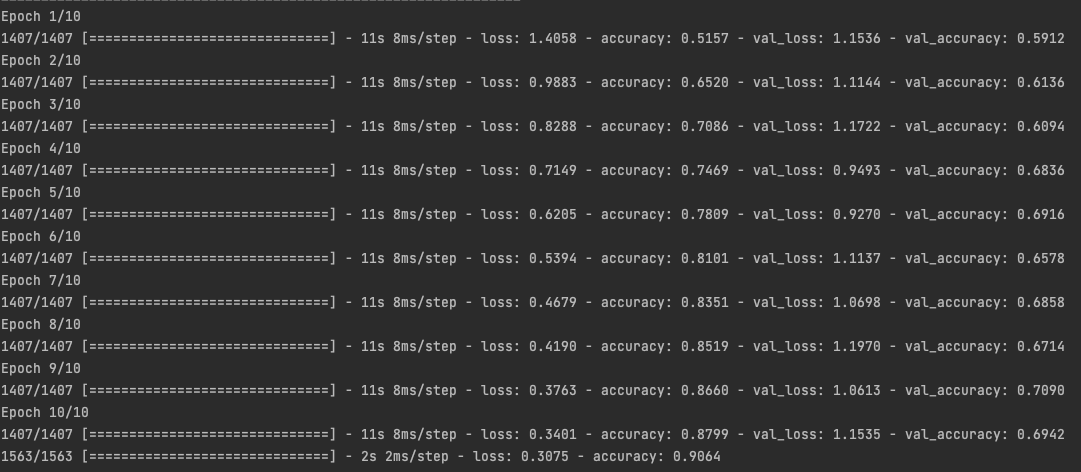
* *validation\_split* – 0,1
* *epochs* – 10
* *learningRate* – 0,001

Chart, line chart

Description automatically generatedChart

Description automatically generated

13 pav. Trečiojo bandymo tiklsumo ir nuostolio grafikas ir matrica



14 pav. Trečiojo bandymo mokymosi rezultatai

Klasifikavimo tikslumas testavimo duomenims gautas 90,64 % tai geriausias rezultatas kuri pavyko gauti iki šio momento.

Ketvirto bandymo parametrai:

* *validation\_split* – 0,1
* *epochs* – 50
* *learningRate* – 0,001

Chart, line chart

Description automatically generatedA picture containing graphical user interface

Description automatically generated

15 pav. Ketvirtojo bandymo tiklsumo ir nuostolio grafikas ir matrica

Text

Description automatically generated

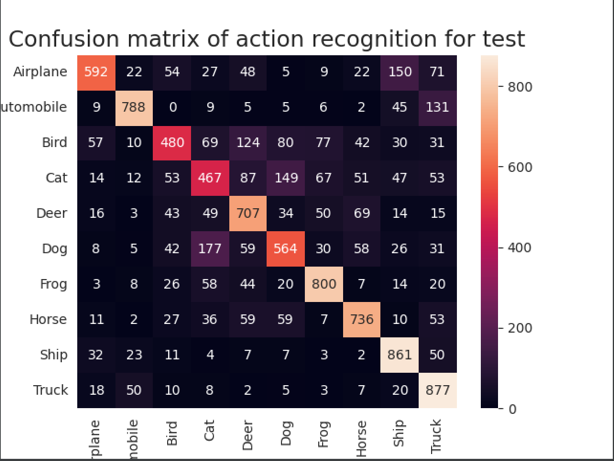
16 pav. Ketvirtojo bandymo mokymosi rezultatai

Klasifikavimo tikslumas testavimo duomenims gautas dar geresnis – 96,47 %. Matome kad keičiant tik epochų skaičių rezultatas gerėja.

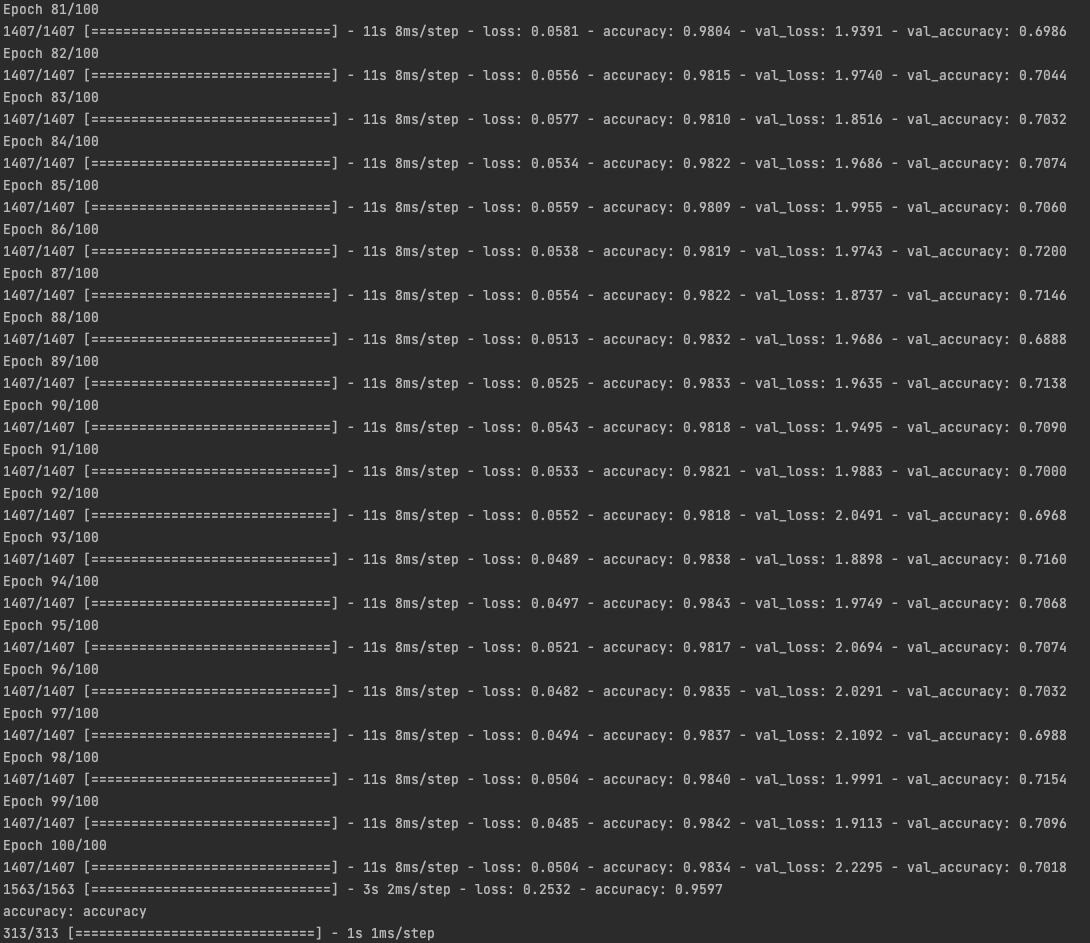
Penkto bandymo parametrai:

* *validation\_split* – 0,1
* *epochs* – 100
* *learningRate* – 0,001

Chart, line chart

Description automatically generated

17 pav. Penktojo bandymo tikslumo ir nuostolio grafikas ir matrica



18 pav. Penktojo bandymo mokymosi rezultatai

Klasifikavimo tikslumas testavimo duomenims gautas – 95,97 %. Bet truputi blogesnis, nei praeitą bandymą.

Geriausias rezultatas kuri gavome - 96,47 % (15 pav.). Tai yra labai aukštas rezultatas dėl kurio galime teigti, kad epochų skaičiaus didinimas lemia tikslesnį klasifikavimą, bet labai aukštas epochų skaičius atvirkščiai gali pabloginti tikslumą kaip matome penktajame bandyme (17 pav.). *Validation\_split* ir *learningRate* reikšmių didinimas neigiamai paveikė klasifikavimo tikslumą (11 pav.)

# 4. Išvados

Darbas atliktas pasitelkiant *tenserflow* atvirosios programinės įrangos biblioteką ir *python* programavimo kalbą. Pasitelkiant keturis interneto šaltinius buvo sukurta programa, kuri naudodama *CIFAR* vaizdų rinkinį klasifikuoją paveiksliukus į 10 klasių. Testuojant keturias tinklo architektūras buvo nustatyta, kad trečioji architektūra su vienu *Dropout* pasiekė 86,18 % tikslumą, esant vienodiems hiperparametrams. Nustačius geriausią architektūrą, buvo nustatyti hiperparametrai su kuriais klasifikavimo tikslumas yra aukščiausias. Esant parametrams:

- *validation\_split* – 0,1

- *epochs* – 50

- *learningRate* – 0,001

Buvo pasiektas aukščiausias rezultatas (15 pav.) 96,47 %. Tolimesnis epochų skaičiaus didinimas rezultatus truputi pablogino (17 pav.), o *learningRate* ir *validation\_split* reikšmių didinimas taipogi neigiamai paveikė klasifikavimo tikslumą. Tikėtina, kad viena iš priežasčių lėmusi neigiamą rezultatą didinant epochų skaičių yra prastas *learningRate* ir *validation\_split* parametrų parinkimas.

# 5. Naudoti šaltiniai

Image Classification using Tensorflow2.0 on CIFAR-10 dataset ([medium.com](https://medium.com/analytics-vidhya/image-classification-using-tensorflow2-0-with-cifar-10-dataset-cc595ceb0082))

Convolutional Neural Network (CNN) ([TensorFlow](https://www.tensorflow.org/tutorials/images/cnn))

Video classification with a 3D convolutional neural network (CNN) ([TensorFlow](https://www.tensorflow.org/tutorials/video/video_classification))

Image classification of CIFAR-10 using TensorFlow ([kaggle.com](https://www.kaggle.com/code/viratkothari/image-classification-of-cifar-10-using-tensorflow/notebook))

# 6. Priedai

## 6.1. Programos kodas

# importing libraries

import tensorflow as tf

from tensorflow import keras

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Flatten, Dense

from tensorflow.keras.layers import BatchNormalization, Conv2D

from tensorflow.keras.layers import MaxPool2D, Activation, MaxPooling2D, Dropout

sns.set\_style('darkgrid')

# loading dataset cifar10 or cifar100

# label\_mode='fine' one of "fine", "coarse"

cifar = keras.datasets.cifar10

(X\_train, y\_train), (X\_test, y\_test) = keras.datasets.cifar10.load\_data()

#checking shape and size

print(X\_train.shape)

print(y\_train.shape)

print(X\_test.shape)

print(y\_test.shape)

# https://medium.com/analytics-vidhya/image-classification-using-tensorflow2-0-with-cifar-10-dataset-cc595ceb0082

# defining labels

label = ['Airplane', 'Automobile', 'Bird', 'Cat', 'Deer',

'Dog', 'Frog', 'Horse', 'Ship', 'Truck']

# rescaling data between 0-1

X\_train = X\_train / 255.0

X\_test = X\_test / 255.0

#

# buiding model

model = Sequential()

model.add(Conv2D(64, activation='relu', kernel\_size=(3, 3)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(BatchNormalization())

model.add(Conv2D(32, activation='relu', kernel\_size=(3, 3)))

model.add(Dropout(0.25))

model.add(BatchNormalization())

model.add(Flatten(input\_shape=(32, 32, 3)))

model.add(Dense(80, activation='relu'))

model.add(Dense(10, activation='softmax'))

# compilation of model categorical\_crossentropy - softmax

model.compile(optimizer=tf.keras.optimizers.Adam(learning\_rate=1e-3), loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

input\_shape = (None, 32, 32, 3)

model.build(input\_shape)

model.summary()

# fitting model

history = model.fit(X\_train, y\_train, epochs=100, verbose=1, shuffle = True,

validation\_split=0.1, use\_multiprocessing = True)

test\_loss, test\_acc = model.evaluate(X\_train, y\_train, return\_dict = True, use\_multiprocessing = True)

print("accuracy:", test\_acc)

# https://www.tensorflow.org/tutorials/video/video\_classification

def plot\_history(history):

fig, (ax1, ax2) = plt.subplots(2)

fig.set\_size\_inches(18.5, 10.5)

# Plot loss

ax1.set\_title('Loss')

ax1.plot(history.history['loss'], label = 'train')

ax1.plot(history.history['val\_loss'], label = 'test')

ax1.set\_ylabel('Loss')

# Determine upper bound of y-axis

max\_loss = max(history.history['loss'] + history.history['val\_loss'])

ax1.set\_ylim([0, np.ceil(max\_loss)])

ax1.set\_xlabel('Epoch')

ax1.legend(['Train', 'Validation'])

# Plot accuracy

ax2.set\_title('Accuracy')

ax2.plot(history.history['accuracy'], label = 'train')

ax2.plot(history.history['val\_accuracy'], label = 'test')

ax2.set\_ylabel('Accuracy')

ax2.set\_ylim([0, 1])

ax2.set\_xlabel('Epoch')

ax2.legend(['Train', 'Validation'])

plt.show()

def get\_actual\_predicted\_labels(dataset1, dataset2):

predicted = model.predict(dataset1)

actual = tf.stack(dataset2, axis=0)

predicted = tf.concat(predicted, axis=0)

predicted = tf.argmax(predicted, axis=1)

return actual, predicted

def plot\_confusion\_matrix(actual, predicted, labels, ds\_type):

cm = tf.math.confusion\_matrix(actual, predicted)

ax = sns.heatmap(cm, annot=True, fmt='g')

sns.set(rc={'figure.figsize':(12, 12)})

sns.set(font\_scale=1.4)

ax.set\_title('Confusion matrix of action recognition for ' + ds\_type)

ax.set\_xlabel('Predicted Action')

ax.set\_ylabel('Actual Action')

plt.xticks(rotation=90)

plt.yticks(rotation=0)

ax.xaxis.set\_ticklabels(labels)

ax.yaxis.set\_ticklabels(labels)

plt.show()

plot\_history(history)

actual, predicted = get\_actual\_predicted\_labels(X\_test, y\_test)

plot\_confusion\_matrix(actual, predicted, label, 'test')

## 6.2. Programinio kodo aprašymas

Pirmoji programos dalis importuoja CIFAR duomenų rinkinį ir padalina jį i mokymosi ir testavimo aibes. Tuomet duomenys yra normalizuojami į intervalą tarp 0 ir 1. Pradedamas modelio kurimas, modeliui nurodomas optimizavimo metodas (“adam”), praradimo funkcijos skaičiavimo algoritmas (“sparse categorical crossentropy”) ir kuri matavima skaičiuoti (“accuracy”). Atspausdinamas sukurto modelio apibendrinimas (“model.summary()”). Modeliui paduodama mokymosi duomenų aibę, epochų skaičiu ir kiti parametrai. Tada jau apmokytas modelis yra testuojamas kaip tiksliai jis priskiria klases. Kitos 3 funkcijos atspausdina grafiką ir klasifikavimo matricą