VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

Vaizdų ir laiko eilučių klasifikavimas naudojant konvoliucinius neuroninius tinklus

Praktinio darbo ataskaita

Atliko: Justinas Bliujus
VU el. p.: justinas.bliujus@mif.stud.vu.lt

Vertino:

Vilnius
2025

1. SKYRIUS

Šio darbo tikslas yra apmokyti konvoliucinius neuroninius tinklus vaizdams ir laiko eilutėms klasifikuoti, atlikti tyrimą pagal duotus nurodymus.

1.1. Naudojami duomenys

1.1.1. Vaizdams

Tyrimui naudojama keksiukų ir šuniukų duomenų aibė iš https://www.kaggle.com/datasets/samuelcortinhas/muffin-vs-chihuahua-image-classification/data.

Ją sudaro 3199 šuniukų vaizdai ir 2718 keksiukų vaizdai. Taigi turime dvi klases. Dėl resursų trūkumo, apmokymui naudojama tik dalis duomenų, tai yra pusė šuniukų vaizdų ir pusė keksiukų vaizdų. Duomenys normalizuojami, pikselių reikšmes sutraukiant į intervalą nuo 0 iki 1.

1.1.2. Laiko eilutėms

Tyrimui naudojama CMU duomenų aibė iš https://www.cs.cmu.edu/~keystroke/.

Tai klavišų paspaudimų aibe, kai renkant slaptažodį „tie5Roanl“ buvo įrašomas klavišų paspaudimų, atleidimų laikas ir laikas tarp klavišų paspaudimų. Vienos laiko eilutės požymių skaičius 31. Buvo 57 dalyviai, kiekvienas iš jų rinko slaptažodį po 50 kartų 8 sesijų metu. Tyrimui pasirinkti 30 pirmųjų naudotojų pirmosios sesijos metu.

1.2. Naudojamų konvoliucinių tinklų architektūros ir parametrai

1.2.1. Pradinė vaizdams naudojama architektūra

Pradinė architektūra susideda iš 7 sluoksnių.

Pirmasis - konvoliucinis 2 dimensijų sluoksnis su 8 filtrais, branduolio ilgis ir aukštis 2, žingsnis taip pat 2. Aktyvacijos funkcija ReLu.

Antras sluoksnis sujungimo - sutraukia duomenų apimtį, paima didžiausią reikšmę iš kiekvieno bloko, kurio ilgis ir aukštis 2.

Trečias sluoksnis toks pats kaip ir pirmasis, bet filtrų yra 16.

Ketvirtas sluoksnis toks pats kaip antrasis.

Penktas sluoksnis paverčia dvimatę matricą į vienmatį vektorių.

Šeštas sluoksnis yra pilnai sujungtas sluoksnis su 256 neuronais, naudoja ReLu aktyvacijos funkciją.

Septintas sluoksnis taip pat pilnai sujungtas bet turi tik 1 neuroną, todėl naudojame sigmoidinę funkciją, kad galėtume klasifikuoti binariškai.

1.2.2. Pradinė laiko eilutėms naudojama architektūra

Susideda iš 7 sluoksnių.

Pirmasis – konvoliucinis vienos dimensijos sluoksnis su 64 filtrais, branduolio dydis 3, žingsnio dydis 1, aktyvacijos funkcija ReLu.

Antrasis sluoksnis – sujungimo, dydis 2.

Trečiasis sluoksnis toks kaip ir pirmasis, bet su 128 filtrais.

Ketvirtasis toks pats kaip antrasis.

Penktasis sluoksnis paverčia konvoliucinius duomenis į vienmatį vektorių.

Šeštasis sluoksnis – pilnai sujungtas sluoksnis su 128 neuronais.

Septintasis sluoksnis – naudoja softmax funkciją, turi 30 neuronų, nes norime gauti 30 klasių.

Epochų skaičius 30, mokymosi greitis 0,001.

1.2.3. Vaizdų duomenų paruošimas

Duomenys jau buvo išskaidyti, todėl visi šuniukų vaizdai sudėti į vieną failą ir visi keksiukų vaizdai sudėti į kitą failą. Duomenys išmaišomi ir kadangi naudojame tik pusę duomenų, apmokymui imami pirmi 40 % duomenų, validavimui sekantys 5 %, testavimui sekantys 5 %.

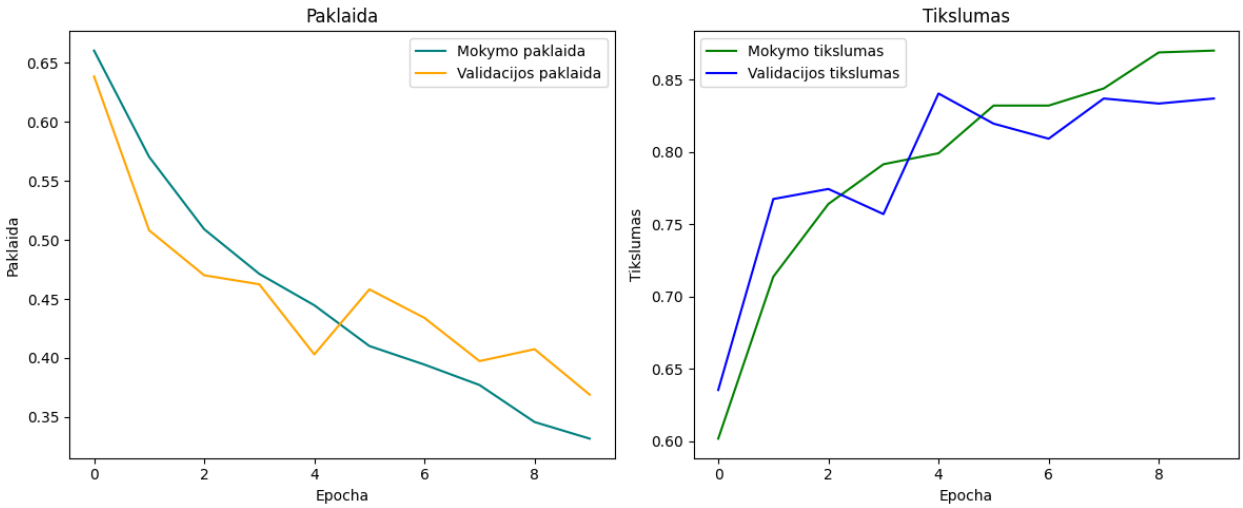
1.2.4. Laiko eilučių duomenų paruošimas

Naudojami visi pasirinktų 30 dalyvių duomenys. Duomenys išmaišomi ir 80 % jų naudojami apmokymui, 10 % validavimui ir 10 % testavimui.

1.3. Tyrimai

1.3.1. Tikslumas ir paklaida keičiant tinklo architektūrą ir parametrus vaizdų atveju

Tikslumas ir paklaida su pirmine architektūra *1* pav.



1 pav. Pirminės architektūros rezultatai

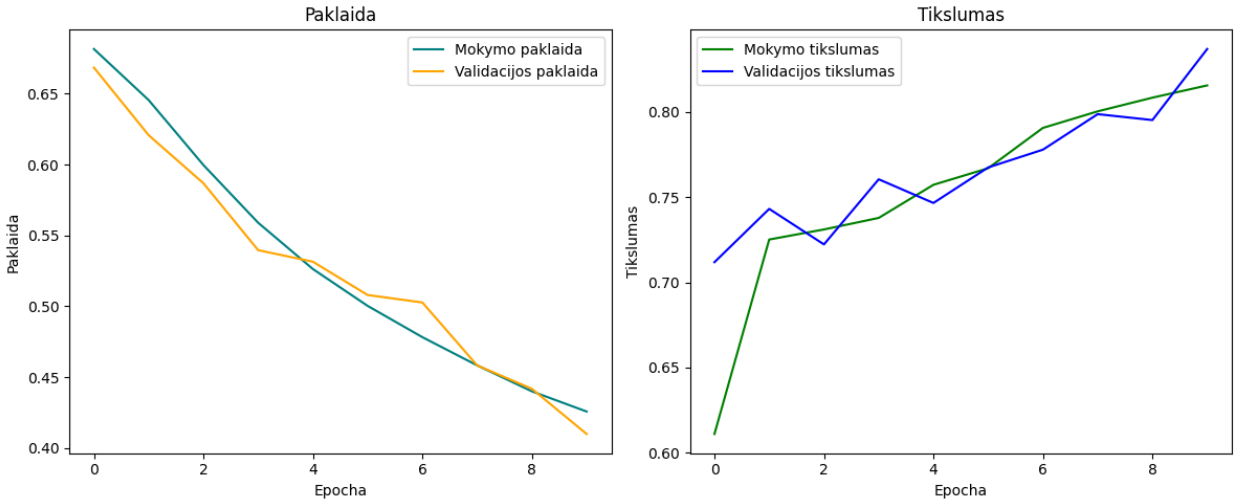
Gauta paklaida mokymo duomenims: 0,3316

Gauta paklaida validavimo duomenims: 0,3690

Gautas tikslumas mokymo duomenims: 0,8699

Gautas tikslumas validavimo duomenims: 0,8368

Pridėjus trečią konvoliucinį sluoksnį su 8 filtrais gauti rezultatai *2* pav.:



2 pav. pirmo bandymo rezultatai

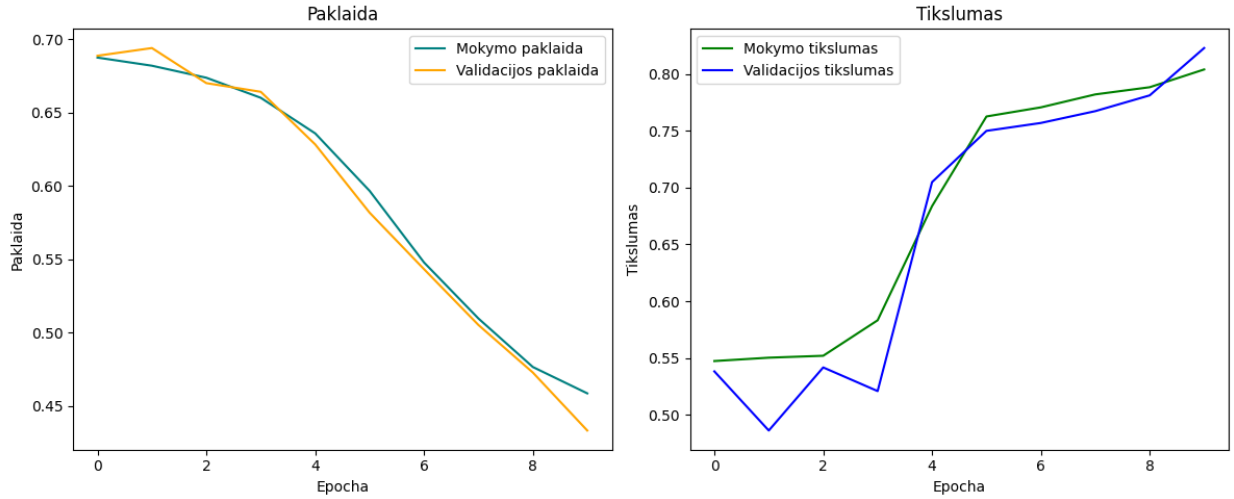
Gauta paklaida mokymo duomenims: 0,4257

Gauta paklaida validavimo duomenims: 0,4099

Gautas tikslumas mokymo duomenims: 0,8155

Gautas tikslumas validavimo duomenims: 0,8368

Šiuo atveju paklaidos padidėjo, todėl kitas bandymas konvoliucinį sluoksnį įterpti kitoje vietoje. Pridėjus trečią konvoliucinį sluoksnį tarp pirmo ir antro su 8 filtrais rezultatai matomi *3* pav.



3 pav. antro bandymo rezultatai

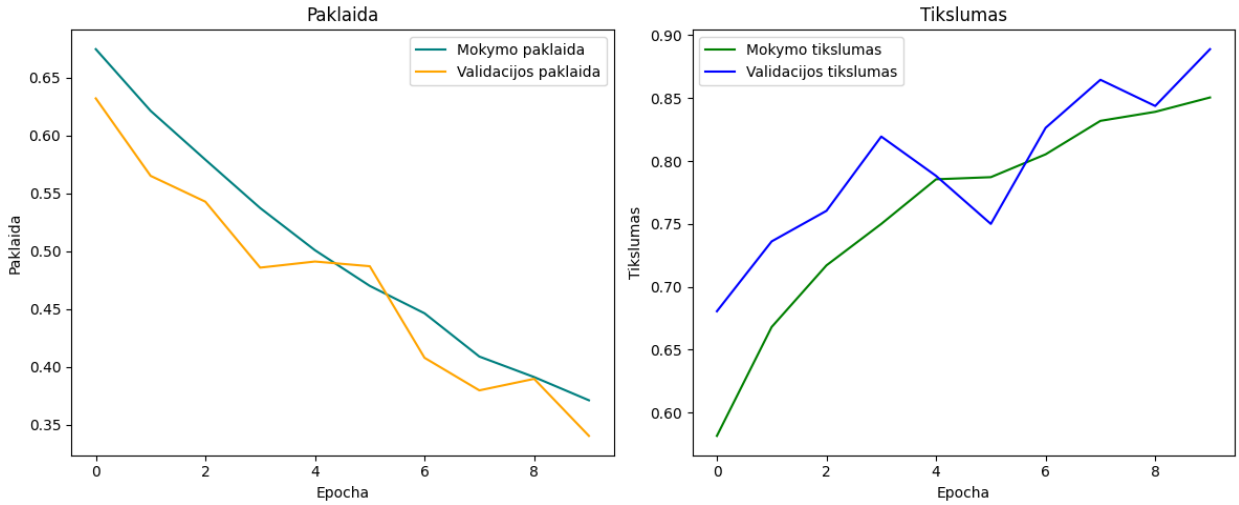
Gauta paklaida mokymo duomenims: 0,4585

Gauta paklaida validavimo duomenims: 0,4331

Gautas tikslumas mokymo duomenims: 0,8041

Gautas tikslumas validavimo duomenims: 0,8229

Gautos paklaidos dar aukštesnės, todėl geriausias variantas kol kas buvo pradinis. Pridėjus išmetimo sluoksnį po pirmo sujungimo sluoksnio, su reikšme 0,4 gauti rezultatai *4* pav.



4 pav. trečio bandymo rezultatai

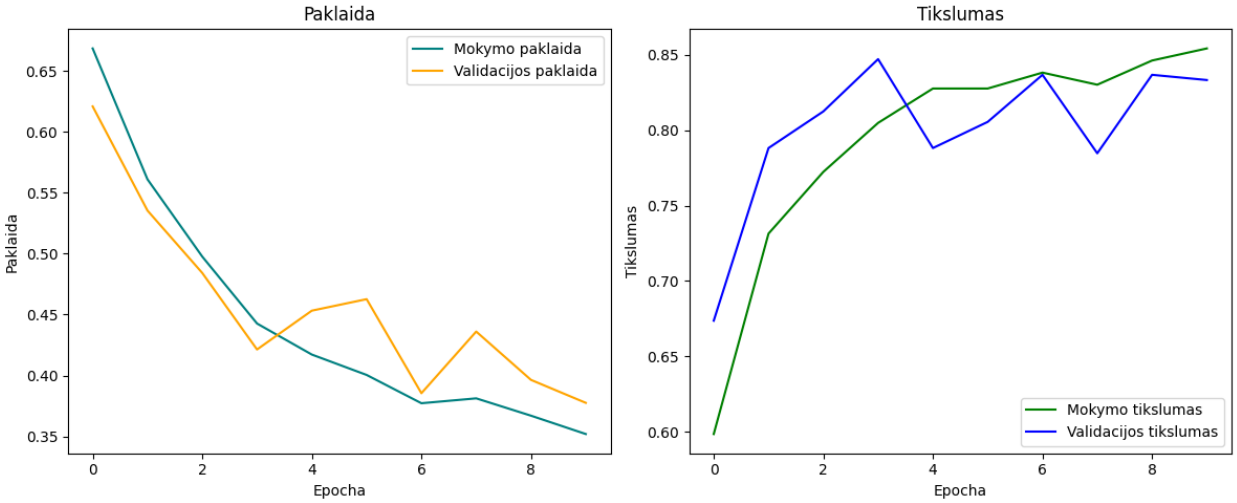
Gauta paklaida mokymo duomenims: 0,3713

Gauta paklaida validavimo duomenims: 0,3406

Gautas tikslumas mokymo duomenims: 0,8505

Gautas tikslumas validavimo duomenims: 0,8889

Gautos reikšmės yra geresnės nei kitų bandymų, bet panašios į pradines. Pridėjus dar vieną išmetimo sluoksnį su reikšme 0,2 po konvoliucinio sluoksnio su 16 filtrų gauti rezultatai *5* pav. taip pat stipriai nesiskiria.



5 pav. ketvirto bandymo rezultatai

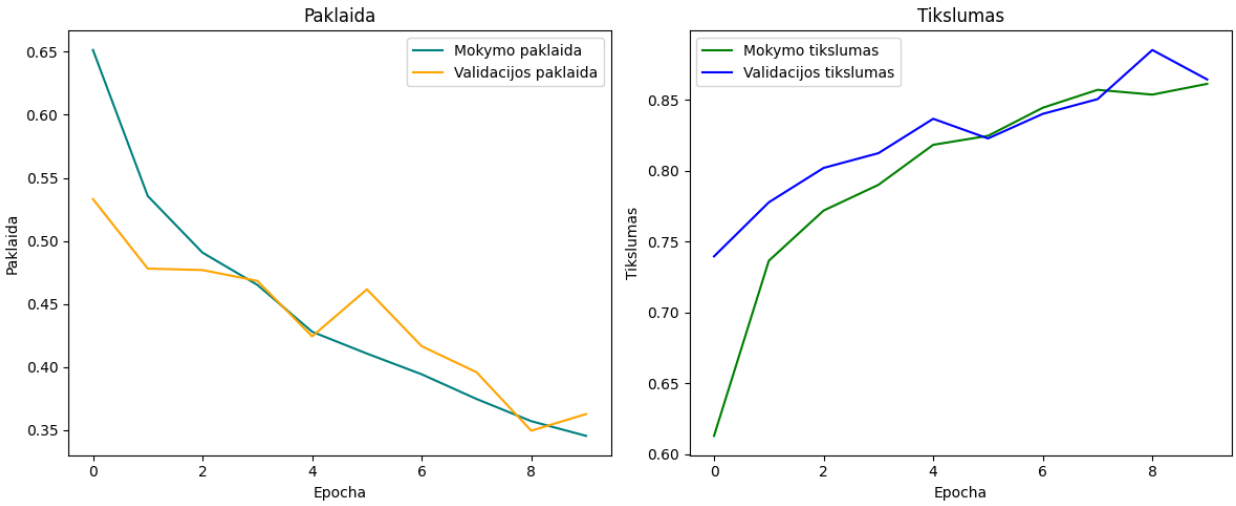
Gauta paklaida mokymo duomenims: 0,3519

Gauta paklaida validavimo duomenims: 0,3775

Gautas tikslumas mokymo duomenims: 0,8543

Gautas tikslumas validavimo duomenims: 0,8333

Bandymas su tanh aktyvacijos funkcija *6* pav. rezultatų nepagerina, kaip ir bandymas su swift akyvacijos funkcija *7* pav.



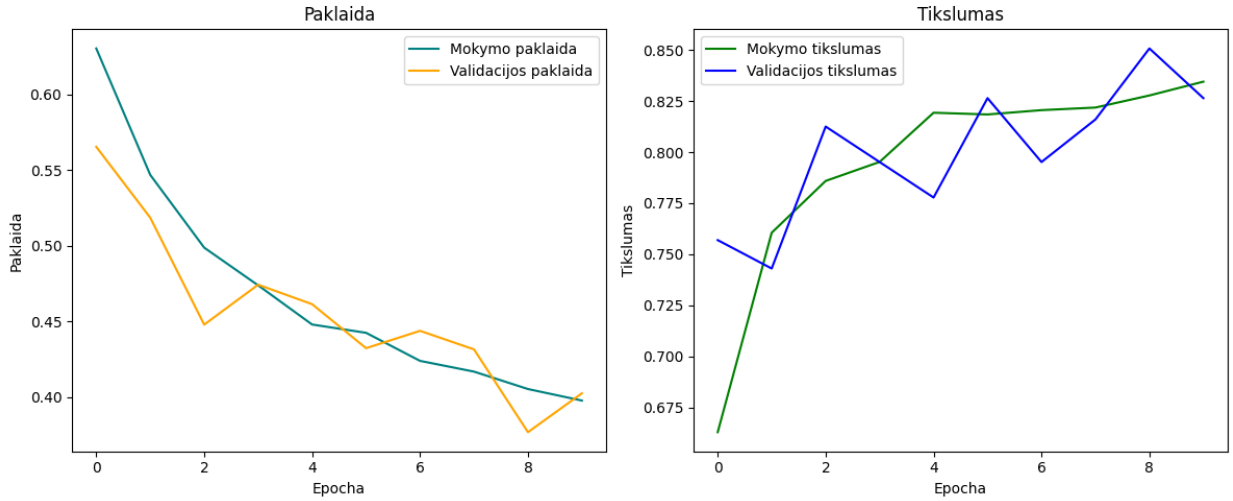
6 pav. penktas bandymas

Gauta paklaida mokymo duomenims: 0,3455

Gauta paklaida validavimo duomenims: 0,3628

Gautas tikslumas mokymo duomenims: 0,8615

Gautas tikslumas validavimo duomenims: 0,8646



7 pav. šeštas bandymas

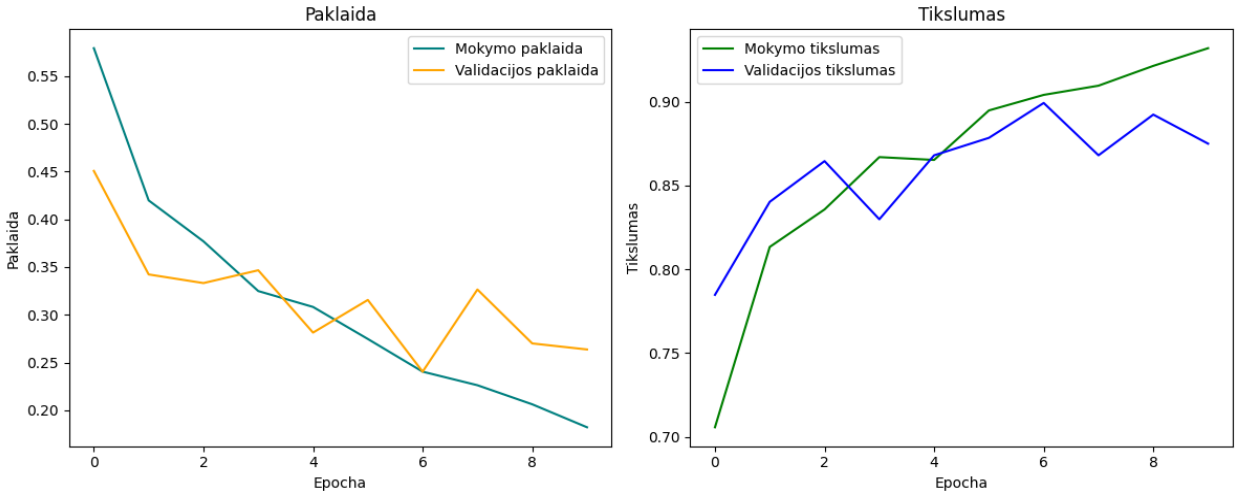
Gauta paklaida mokymo duomenims: 0,3976

Gauta paklaida validavimo duomenims: 0,4025

Gautas tikslumas mokymo duomenims: 0,8345

Gautas tikslumas validavimo duomenims: 0,8264

Kadangi bandymai su kitomis aktyvacijos funkcijomis įtakos nepadarė, grįžtama prie ReLu. Bandymas keisti mokymosi greitį nuo 0,0001 iki 0,001 pagerino rezultatus ir sumažino paklaidas *8* pav.



8 pav. septintas bandymas

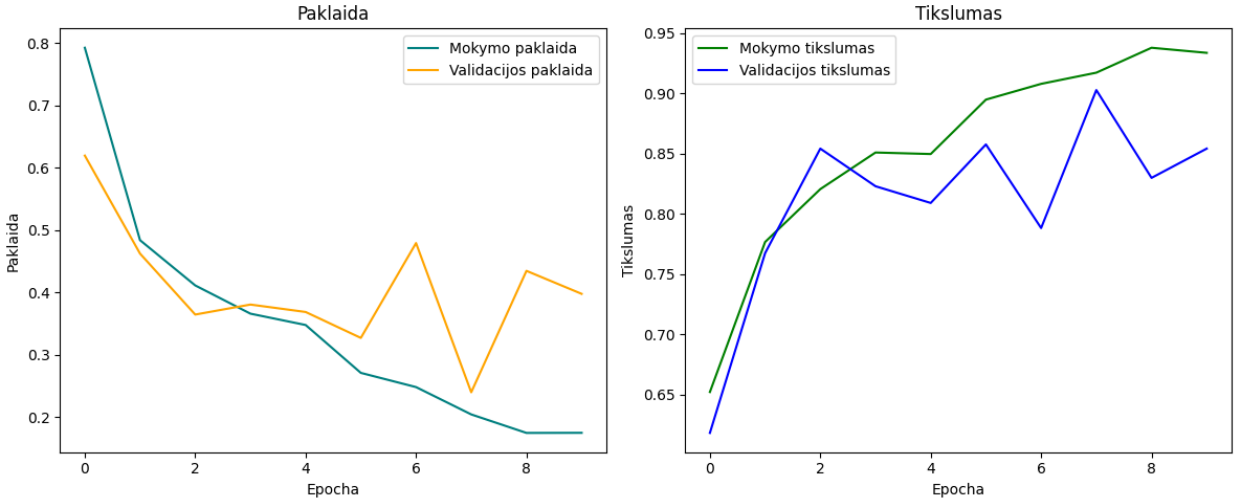
Gauta paklaida mokymo duomenims: 0,1822

Gauta paklaida validavimo duomenims: 0,2637

Gautas tikslumas mokymo duomenims: 0,9320

Gautas tikslumas validavimo duomenims: 0,8750

Bandymas mokymosi greitį padidinti iki 0,01 rezultatus pablogino, nes padidėjo validacijos paklaida *9* pav.



9 pav. aštuntas bandymas

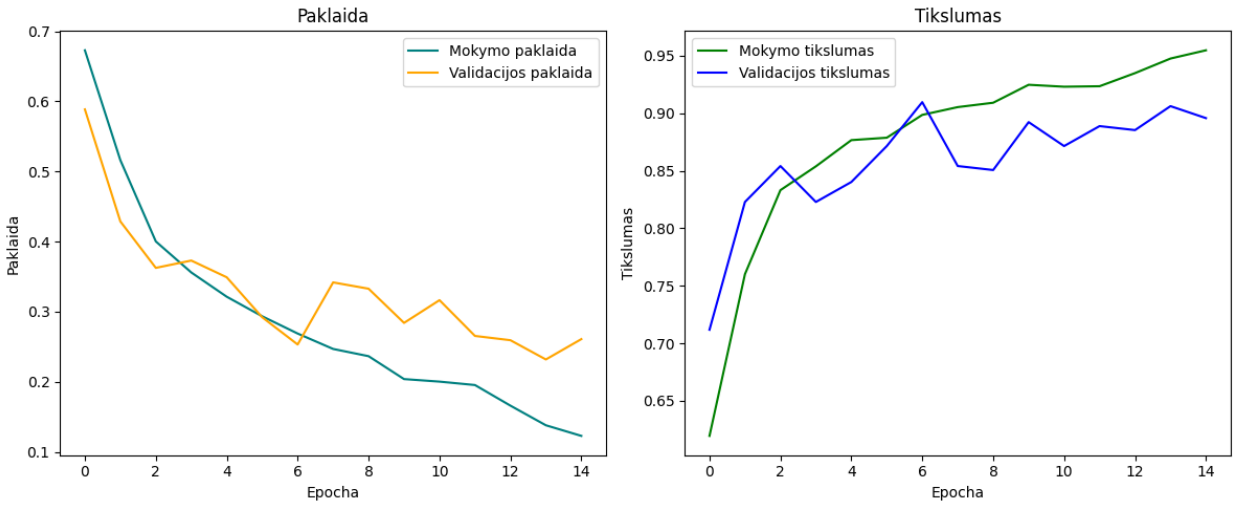
Gauta paklaida mokymo duomenims: 0,1748

Gauta paklaida validavimo duomenims: 0,3976

Gautas tikslumas mokymo duomenims: 0,9337

Gautas tikslumas validavimo duomenims: 0,8542

Bandymas su ReLu funkcija, kaimokymosi greitis 0,001, epochų skaičius 15. Rezultatai tokie kaip ir septintame bandyme, bet validacijos ir mokymo paklaidos stipriau išsiskiria.



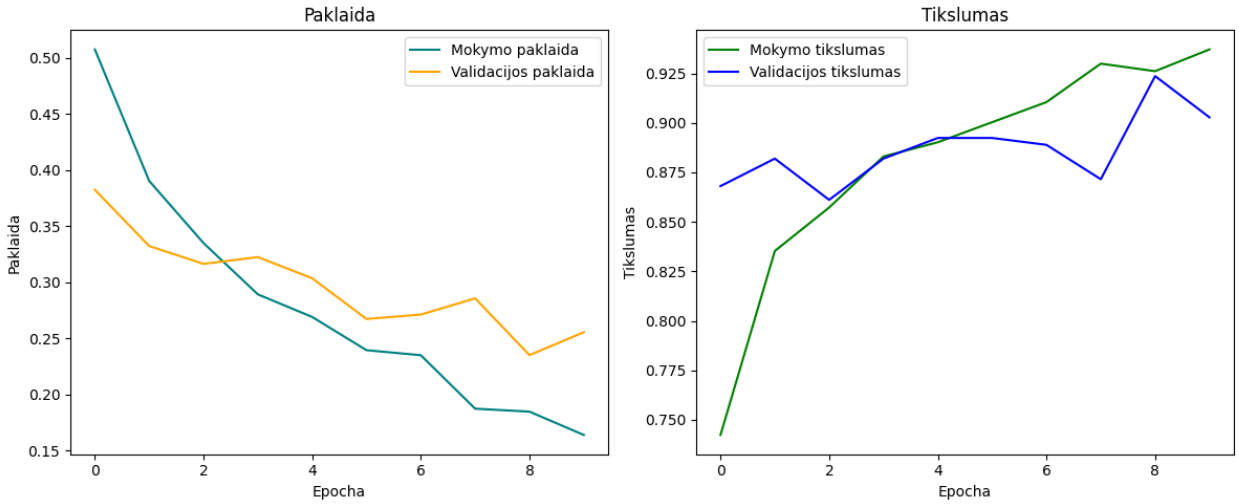
Gauta paklaida mokymo duomenims: 0,1228

Gauta paklaida validavimo duomenims: 0,2609

Gautas tikslumas mokymo duomenims: 0,9548

Gautas tikslumas validavimo duomenims: 0,8958

Siekiant sumažinti skirtumą tarp validavimo ir mokymo paklaidos grįžtama prie 10 epochų. Išmetimo sluoksniui po konvoliucijos su 16 filtrų suteikiama 0,4 reikšmė, išmetimo sluoksniui po pilnai sujungto sluoksnio suteikiama reikšmė 0,2. Rezultatai *10* pav. geriausi, nes pasiektas tikslumas virš 90 % tiek validavimo tiek mokymo duomenims, pasiekta viena žemiausių paklaidų.



10 pav. devintas bandymas

Gauta paklaida mokymo duomenims: 0,1641

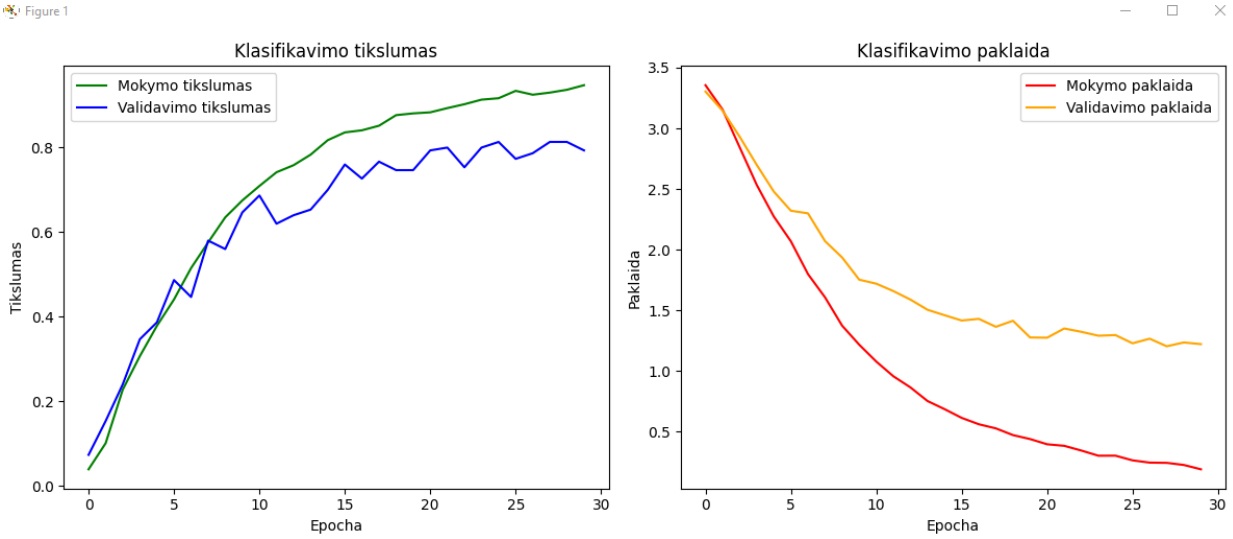
Gauta paklaida validavimo duomenims: 0,2557

Gautas tikslumas mokymo duomenims: 0,9371

Gautas tikslumas validavimo duomenims: 0,9028

1.3.2. Tikslumas ir paklaida keičiant architektūrą ir parametrus laiko eilučių atveju

Rezultatai gauti su pradiniais duomenimis *11* pav. Gauta validavimo paklaida daug didesnė nei apmokymo paklaida.



11 pav. Rezultatai su pradiniais duomenimis

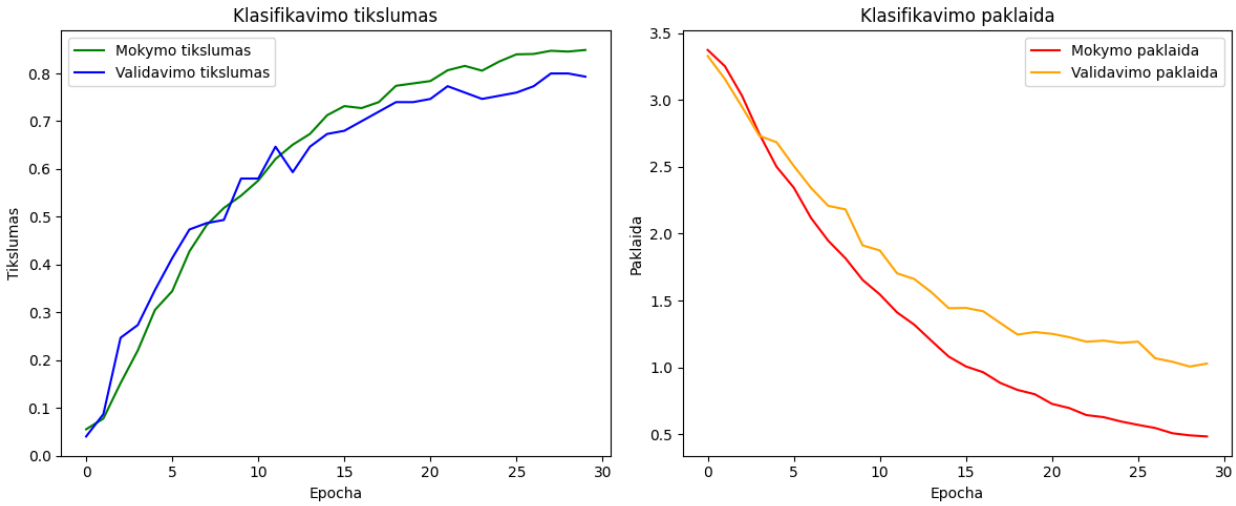
Mokymo paklaida: 0,1908

Validavimo paklaida: 1,2223

Mokymo tikslumas: 0,9475

Validavimo tikslumas: 0,7933

Pirmas bandymas – po pirmų dvejų sluoksnių įterpti išmetimo sluoksnį su reikšme 0,4. Gauta validavimo paklaida didelė, bet mažesnė nei buvo (*12* pav.).



12 pav. Pirmasis bandymas

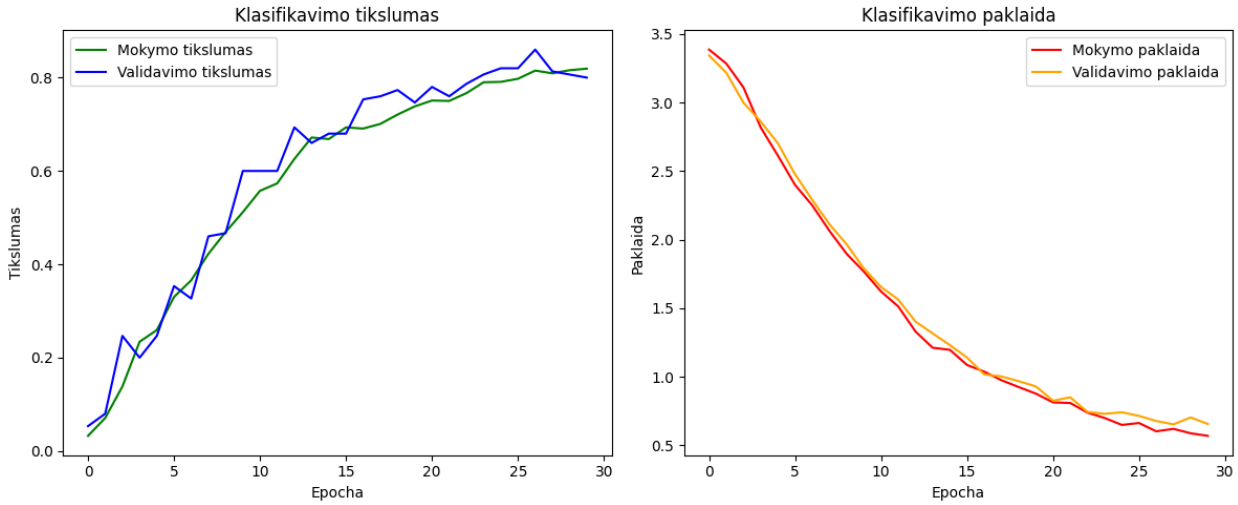
Mokymo paklaida: 0,4843

Validavimo paklaida: 1,0289

Mokymo tikslumas: 0,8492

Validavimo tikslumas: 0,7933

Antrasis bandymas – įterpimas dar vieno išmetimo sluoksnio, po antro konvoliucijos etapo. *13* pav. rezultatuose matoma, kad validavimo ir mokymo paklaidos supanašėjo.



13 pav. Antrasis bandymas

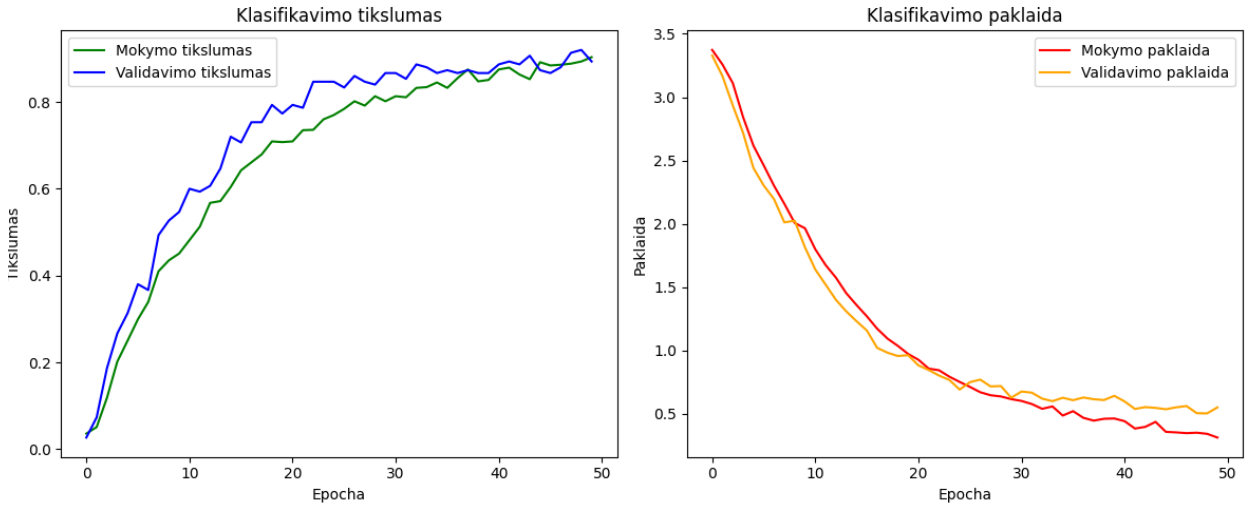
Mokymo paklaida: 0,5682

Validavimo paklaida: 0,6548

Mokymo tikslumas: 0,8192

Validavimo tikslumas: 0,8000

Trečio bandymo metu padidintas epochų skaičius iki 50. *14* pav. matomas didesnis tikslumas ir žemesnės paklaidos. Tai geriausias variantas.



14 pav. Trečias bandymas

Mokymo paklaida: 0,3133

Validavimo paklaida: 0,5515

Mokymo tikslumas: 0,9033

Validavimo tikslumas: 0,8933

1.4. Geriausias atvejis

1.4.1. Geriausio atvejo vaizdų apdorojime parametrai

Naudojama ReLu aktyvacijos funkcija, mokymosi greitis 0,001, epochų skaičius 10.

Architektūra: konvoliucinis sluoksnis su 8 filtrais, 2 aukščio ir 2 pločio, žingsnis 2. Sujungimo sluoksnis, kai aukštis 2 ir plotis 2. Konvoliucinis sluoksnis su 16 filtrų, 2 aukščio ir 2 pločio, žingsnis 2. Išmetimo sluoksnis su reikšme 0,4. Sujungimo sluoksnis 2 aukščio ir 2 pločio. Ištiesinimo sluoksnis. Pilnai sujungtas sluoksnis. Išmetimo sluoksnis su reikšme 0,2. Pilnai sujungtas sluoksnis su sigmoidine aktyvacijos funkcija.

1.4.2. Geriausio atvejo laiko eilučių apdorojime parametrai

Naudojama ReLu aktyvacijos funkcija, paskutiniame pilnai sujungtame sluoksnyje softmax funkcija.

Pirmasis – konvoliucinis vienos dimensijos sluoksnis su 64 filtrais, branduolio dydis 3, žingsnio dydis 1, aktyvacijos funkcija ReLu.

Antrasis sluoksnis – sujungimo, dydis 2.

Trečiasis sluoksnis išmetimo su reikšme 0,4.

Ketvirtasis sluoksnis toks kaip ir pirmasis, bet su 128 filtrais.

Penktasis toks pats kaip antrasis.

Šeštasis sluoksnis išmetimo su reikšme 0,4.

Septintasis sluoksnis paverčia konvoliucinius duomenis į vienmatį vektorių.

Aštuntasis sluoksnis – pilnai sujungtas sluoksnis su 128 neuronais.

Devintasis sluoksnis – naudoja softmax funkciją, turi 30 neuronų, nes norime gauti 30 klasių.

Epochų skaičius 50, mokymosi greitis 0,001.

1.5. Testavimo aibės rezultatai

1.5.1. Vaizdų atveju

Gauta paklaida mokymo duomenims: 0,1641

Gauta paklaida validavimo duomenims: 0,2557

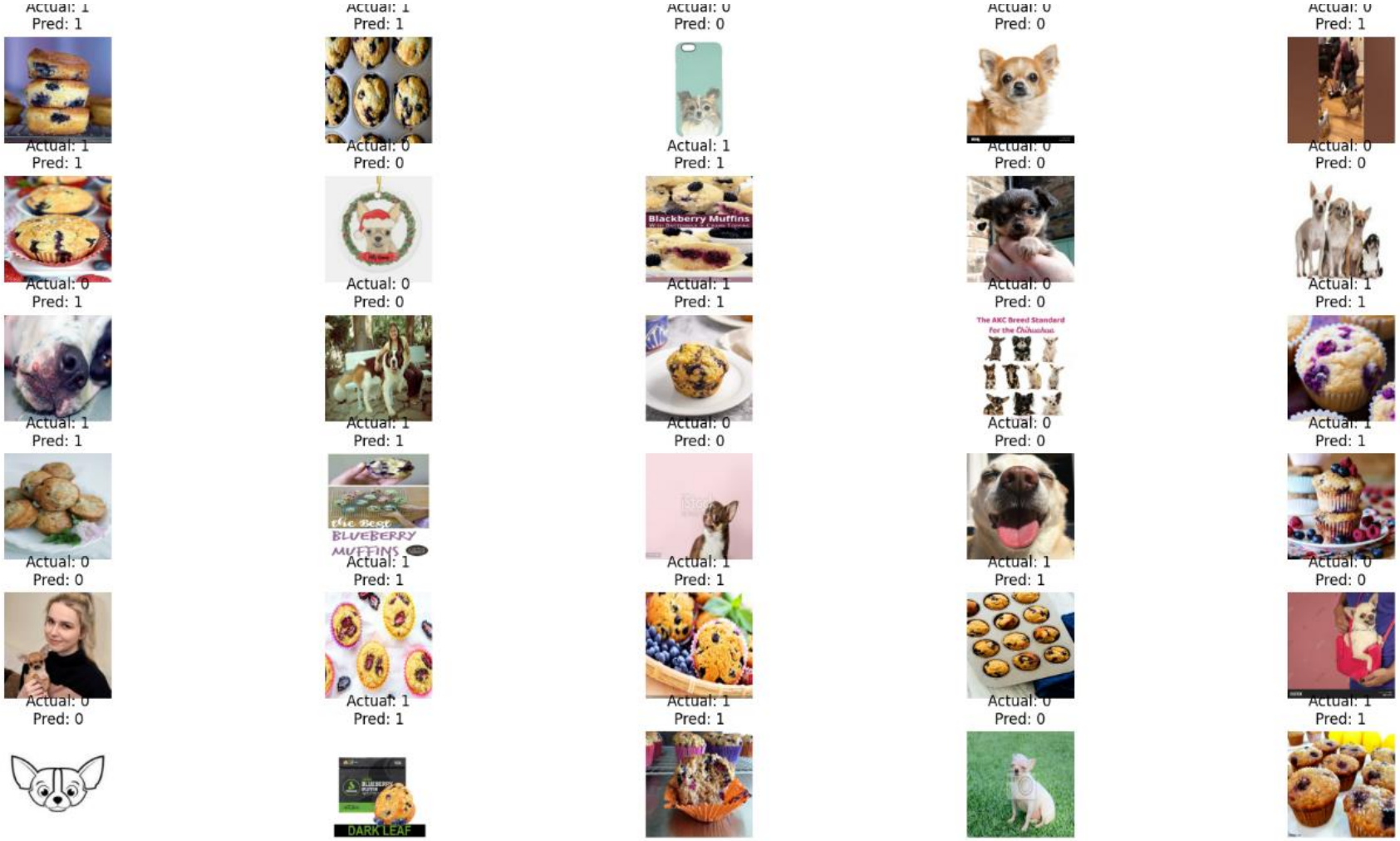
Gautas tikslumas mokymo duomenims: 0,9371

Gautas tikslumas validavimo duomenims: 0,9028

Klasifikavimo matrica 1 Lentelė parodo, kad 262 kartus klasės buvo priskirtos teisingai. 5 kartus 0 klasė (šuniukų) buvo priskirta keksiukams ir 21 kartą keksiukų klasė buvo priskirta šuniukams. Taigi modeliui lengviau priskirti reikiamą klasę kai nuotraukoje šuniukas, nei keksiukas. *15* pav. nurodyta 30 paskutinių klasifikavimų, su prognozuotomis ir tikromis klasėmis bei vaizdais. Buvo suklysta 2 kartus iš 30.

1 Lentelė. Klasifikavimo matrica

	Prognozuotos	
Tikros	***0***	***1***
0	134	21
1	5	128



15 pav. dalis testavimo duomenų klasifikacijos

1.5.2. Laiko eilučių atveju

Mokymo paklaida: 0,3914

Validavimo paklaida: 0,4693

Mokymo tikslumas: 0,8817

Validavimo tikslumas: 0,8533

2 Lentelė parodo kaip buvo klasifikuojami duomenys. Iš 30 rezultatų, 8 buvo klasifikuojami neteisingai, 22 teisingai.

2 Lentelė. 30 bandymų su testiniais duomenimis

Indeksas	***Tikra klasė***	***Prognozė***
0	20	20
1	16	16
2	28	9
3	3	3
4	6	6
5	25	25
6	12	10
7	15	24
8	22	22
9	13	13
10	0	0
11	24	24
12	29	29
13	2	2
14	17	17
15	4	4
16	26	26
17	5	5
18	14	11
19	7	7
20	27	27
21	10	10
22	1	3
23	18	18
24	9	9
25	19	19

26	8	7
27	23	10
28	21	7
29	11	11

1.6. Išvados

Buvo nustatyta, kad išmetimo sluoksniai padeda sumažinti paklaidą kai modelis persimoko.

Nors tiek vaizdų tiek laiko eilučių apdorojime pasiektas didesnis nei 85 % tikslumas, laiko eilučių apdorojime yra aukšta paklaida, todėl modelis nėra toks tikslus.

Vaizdų modelio apmokymas reikalauja daugiau skaičiavimo resursų, tačiau laiko eilučių modelis turi daug daugiau klasių, dėl to yra sudėtingesnis klasifikavimo atžvilgiu.

1 Priedas. Vaizdų apdorojimo programinis kodas

```
import tensorflow as tf
from matplotlib import pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
from tensorflow.keras.metrics import BinaryAccuracy
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dropout
from tensorflow.keras.activations import swish
import numpy as np

data = tf.keras.utils.image_dataset_from_directory("data", batch_size=32) #duomenys
cia sumaisomi ir paimami

data = data.map(lambda x, y: (x / 255, y)) #normalizavimas padalinant iš 255
iterator = data.as_numpy_iterator()
batch = iterator.next()

#duomenu padalijimas
# 40% train, 5% val, 5% test, tai santykis 8:1:1
train_size = int(len(data) * 0.4)
val_size = int(len(data) * 0.05)
test_size = int(len(data) * 0.05)
train = data.take(train_size)
val = data.skip(train_size).take(val_size)
test = data.skip(train_size + val_size).take(test_size)
#print(len(train), len(val), len(test))

#apmokymas
model = Sequential() #filtrai,kernel,zingsnis         256x256x3rgb
model.add(Conv2D(8, (2,2), 2, activation='relu', input_shape=(256, 256, 3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(16, (2,2), 2, activation='relu'))
model.add(Dropout(0.4))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())

model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
#loss = binary_crossentropy arba hinge
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy',
metrics=['accuracy'])
model.summary()

logdir="logs"
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
history = model.fit(train, epochs=10, validation_data=val,
callbacks=[tensorboard_callback])

#grafikai
fig, axs = plt.subplots(1, 2, figsize=(12, 5))
# Paklaidos grafikas
axs[0].plot(history.history['loss'], label='Mokymo paklaida', color='teal')
axs[0].plot(history.history['val_loss'], label='Validacijos paklaida', color='orange')
axs[0].set_title('Paklaida')
axs[0].set_xlabel('Epocha')
axs[0].set_ylabel('Paklaida')
axs[0].legend()
# Tikslumo grafikas
axs[1].plot(history.history['accuracy'], label='Mokymo tikslumas', color='green')
```

```
axs[1].plot(history.history['val_accuracy'], label='Validacijos tikslumas',
color='blue')
axs[1].set_title('Tikslumas')
axs[1].set_xlabel('Epocha')
axs[1].set_ylabel('Tikslumas')
axs[1].legend()
plt.tight_layout()
plt.show()
final_train_loss = history.history['loss'][-1]
final_val_loss = history.history['val_loss'][-1]
final_train_acc = history.history['accuracy'][-1]
final_val_acc = history.history['val_accuracy'][-1]
print(f"Mokymo paklaida: {final_train_loss:.4f}")
print(f"Validacijos paklaida: {final_val_loss:.4f}")
print(f"Mokymo tikslumas: {final_train_acc:.4f}")
print(f"Validacijos tikslumas: {final_val_acc:.4f}")

#modelio ivertinimas su testiniais duomenimis
acc = BinaryAccuracy()
y_true = []
y_pred = []
last_x_batch = None

for batch in test.as_numpy_iterator():
    x, y = batch
    last_x_batch = x
    predicted = model.predict(x)
    preds = np.round(predicted).astype(int)
    y_true.extend(y)
    y_pred.extend(preds)
#30 vaizdu klasifikavimo vizualizavimas
last_images = last_x_batch[-30:]
last_true_labels = y_true[-30:]
last_pred_labels = y_pred[-30:]
plt.figure(figsize=(15, 15))
for i in range(30):
    plt.subplot(6, 5, i + 1)
    plt.imshow(last_images[i].astype("float32"))
    plt.title(f"Actual: {int(last_true_labels[i])}\nPred: {last_pred_labels[i]}")
    plt.axis('off')
plt.tight_layout()
plt.show()
```

2 Priedas. Laiko eilučių apdorojimo programinis kodas

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Conv1D, MaxPooling1D, Flatten
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
from tensorflow.keras.optimizers import Adam
import random

data = pd.read_csv("SPD.csv")
# pasirenkami pirmi 30 dalyviu
selected_users = data['subject'].unique()[:30]
data = data[data['subject'].isin(selected_users)]
# pasirenkama pirma sesija
data = data[data['sessionIndex'] == 1]
# isimami nepozymiu stulpeliai
X = data.drop(columns=['subject', 'sessionIndex', 'rep'])

# pervadina klases is s002,s003... i 0,1,...
le = LabelEncoder()
```

```
y = le.fit_transform(data['subject'])

# 0  i (1,0,0,0,0,...,0,0), 1  i (0,1,0,0,0,...,0,0)...
y_categorical = to_categorical(y, num_classes=30)

X_array = X.values

# sumaisymas
indices = np.arange(len(X_array))
np.random.shuffle(indices)
X_array = X_array[indices]
y_categorical = y_categorical[indices]

# 8:1:1 duomenu padalijimas
total_size = len(X_array)
train_size = int(total_size * 0.8)
val_size = int(total_size * 0.1)

X_train = X_array[:train_size]
y_train = y_categorical[:train_size]

X_val = X_array[train_size:train_size + val_size]
y_val = y_categorical[train_size:train_size + val_size]

X_test = X_array[train_size + val_size:]
y_test = y_categorical[train_size + val_size:]

# auto, pozymiai (31), 1 kanalas
X_train = X_train.reshape((-1, X_train.shape[1], 1))
X_val = X_val.reshape((-1, X_val.shape[1], 1))
X_test = X_test.reshape((-1, X_test.shape[1], 1))

# apmokymas
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=3, strides=1, activation='relu',
input_shape=(X_train.shape[1], 1)))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.4))

model.add(Conv1D(filters=128, kernel_size=3, strides=1, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.4))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(30, activation='softmax'))

learning_rate = 0.001
optimizer = Adam(learning_rate=learning_rate)
model.compile(optimizer=optimizer,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()

history = model.fit(X_train, y_train,
                    validation_data=(X_val, y_val),
                    epochs=50, batch_size=32)

# grafiku vaizdavimas
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Mokymo tikslumas', color='green')
plt.plot(history.history['val_accuracy'], label='Validavimo tikslumas', color='blue')
plt.title('Klasifikavimo tikslumas')
plt.xlabel('Epocha')
```

```
plt.ylabel('Tikslumas')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Mokymo paklaida', color='red')
plt.plot(history.history['val_loss'], label='Validavimo paklaida', color='orange')
plt.title('Klasifikavimo paklaida')
plt.xlabel('Epocha')
plt.ylabel('Paklaida')
plt.legend()
plt.tight_layout()
plt.show()

final_train_loss = history.history['loss'][-1]
final_val_loss = history.history['val_loss'][-1]
final_train_acc = history.history['accuracy'][-1]
final_val_acc = history.history['val_accuracy'][-1]

print(f"Mokymo paklaida: {final_train_loss:.4f}")
print(f"Validavimo paklaida: {final_val_loss:.4f}")
print(f"Mokymo tikslumas: {final_train_acc:.4f}")
print(f"Validavimo tikslumas: {final_val_acc:.4f}")

# ivertinimas su testiniais duomenimis
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f"Testavimo paklaida: {test_loss:.4f}")
print(f"Testavimo tikslumas: {test_accuracy:.4f}")
# tikros ir nustatytos klases
true_classes = np.argmax(y_test, axis=1)
y_pred = model.predict(X_test)
predicted_classes = np.argmax(y_pred, axis=1)
# paima po 1 irasa is kiekvienos klases lentelei
samples_per_class = {}
for idx, label in enumerate(true_classes):
    if label not in samples_per_class:
        samples_per_class[label] = idx
    if len(samples_per_class) == 30:
        break

if len(samples_per_class) < 30:
    additional_indices = list(set(range(len(true_classes))) -
set(samples_per_class.values()))
    random.shuffle(additional_indices)
    for idx in additional_indices:
        samples_per_class[true_classes[idx]] = idx
        if len(samples_per_class) == 30:
            break

print(f"{'Indeksas':<8} {'Tikr. klase':<12} {'Prognoze':<10}")
print("-" * 32)
for i, idx in enumerate(samples_per_class.values()):
    true_class = true_classes[idx]
    predicted_class = predicted_classes[idx]
    print(f"{idx:<8} {true_class:<12} {predicted_class:<10}")
```

LITERATŪRA

N. Renotte, „*Build a Deep CNN Clasifier with ANY Images*“, YouTube, įkelta Nicholas Renotte, kovo 25, 2022. [Online Video]. Prieinama https://www.youtube.com/watch?v=jztwpsIzEGc&t=4138s