

**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU  
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I  
INFORMACIJSKIH TEHNOLOGIJA**

**Raspoznavanje uzoraka i strojno učenje**

**Klasifikacija glasovnih naredbi**

**Projektni zadatak**

Ivana Ribičić

Ilija Jazvić

**Osijek, 2024. godina**

# Uvod

Zadatak je bio izraditi klasifikator glasovnih naredbi u Pythonu pomoću dostupnih skupova podataka na internetu (npr. speech command dataset). Evaluirati rješenje na vlastito prikupljenim zvučnim signalima. Izraditi odgovarajuću aplikaciju (aplikacija u streamlitu koja omogućava klasifikaciju zvučnog zapisa ili android aplikaciju koja omogućuje snimanje i klasifikaciju zvučnog signala izravno s mikrofona mobitela ili upravljanje zvučnim naredbama nekim softwareom na računalu).

Na temelju podataka koji predstavljaju glasovne naredbe u obliku zvučnih datoteka potrebno je izraditi klasifikator koji će prepoznati zadane riječi, to jest naredbe.

Ovakav tip sustava može se koristiti u stvarnom životu pojedinca s obzirom da smo okruženi raznim pametnim sustavima od kojih pojedine koristimo za neizbježne radnje s kojima se susrećemo svaki dan.

S obzirom da je klasifikacija govora čest problem u području obrade prirodnog jezika i strojnog jezika, postoji više od jednog pristupa rješavanju tog problema, a neki su navedeni u nastavku.

Za rješavanje navedenog problema mogu se koristiti tradicionalni algoritmi strojnog učenja kao što Gaussovi miješani modeli – GMM (engl. *Gaussian Mixture Models*), skriveni Markovljevi modeli (engl. *Hidden Markov Models*) ili strojevi s potpornim vektorima - SVM (engl. *Vector Support Machines*). SVM-ovi se mogu koristiti za binarnu ili višeklasnu klasifikaciju i primijenjeni su na razne zadatke povezane s govorom.

Također se koriste i razne arhitekture dubokog učenja. Konvolucijske neuronske mreže – CNN (engl. *Convolutional Neural Networks*) mogu se primijeniti na slike spektrograma audio signala u svrhu prepoznavanja lokalnih obrazaca i značajki za klasifikaciju govora. Povratne neuronske mreže – RNN (engl. *Recurrent Neural Networks*), posebice modeli uz dugoročnu kratkotrajnu memoriju (engl. *Long Short-Term Memory – LSTM*), su prikladni za rad sa sekvencijalnim podacima poput audio signala vremenskih nizova. Također postoje i CNN-RNN hibridi koji kombiniraju prednosti CNN-a i RNN-a, gdje CNN-ovi mogu uhvatiti prostorne značajke iz audio reprezentacije, a RNN-ovi obrađuju vremenske ovisnosti. Koriste se i unaprijed istrenirani modeli (engl. *pretrained models*) na velikim skupovima audio podataka (npr. korištenje modela obučenih za općenite audio zadatke) te se podešavaju (engl. *fine tuning*) za specifične zadatke klasifikacije govora što može biti učinkovito kada su podaci ograničeni. [1]

Za rješavanje zadanog problema projektnog zadatka izabran je pristup rješavanja problema klasifikacije korištenjem konvolucijske neuronske mreže.

## Opis korištenih skupova podataka

Za potrebe rješavanja projektnog zadatka bio je potreban skup podataka koji se sastoji od glasovnih naredbi koje su spremljene u obliku zvučnih datoteka.

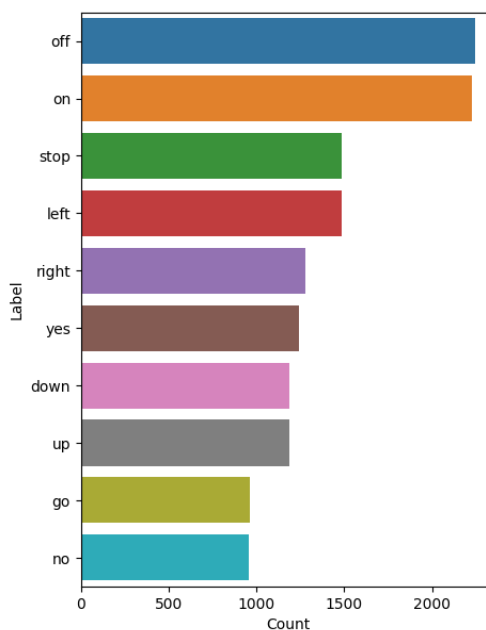
Korišteni podatkovni skup preuzet je s web sjedišta *kaggle.com*. Riječ je o sintetičkim podacima *augmented\_dataset* koji su dobiveni korištenjem programa *espeak* koji pretvara tekst u govor. Prilikom generiranja riječi mijenjali su se način izgovaranja, naglasak pojedinih dijelova riječi, ton, brzina i „govornik“. Riječi su generirane na engleskom jeziku. U podatkovnom skupu dostupna je i verzija *augmented\_dataset\_verynoisy* u kojoj su na svaku riječ dodani pozadinski zvukovi koji djeluju kao smetnja u obliku galame na aerodromu, ulici, željezničkom kolodvoru te sintetički kreiranih smetnji u obliku valova oceana, bijelih šumova i drugih.

Podatkovni skup sastoji se od 30 različitih riječi podijeljenih u 30 mapa u kojoj je za svaku dostupan veći broj primjera u obliku zvučnih datoteka.

Riječi koje se nalaze u skupu su: bed, bird, cat, dog, down, eight, five, four, go, happy, house, left, marvel, nine, no, off, on, one, right, seven, sheila, six, stop, three, tree, two, up, wow, yes, zero.

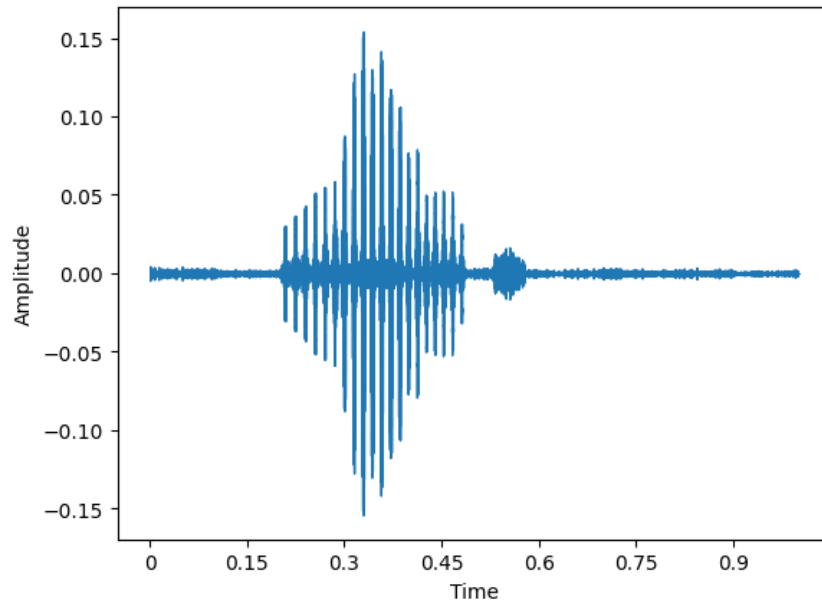
Sve datoteke su spremljene u .wav formatu (16 bit po uzorku, mono – jedan kanal, frekvencija 16000 Hz) u trajanju od jedne sekunde (1s). [2]

Proučavanjem sadržaja dostupnog skupa podataka odlučeno je koristiti 10 različitih riječi koje mogu predstavljati naredbe za upravljanje određenim sustavom. To su: down, go, left, no, off, on, right, stop, up, yes.



Slika 1 Graf prikaz podatkovnog skupa

Korišteni skup podataka nije zahtijevao opsežnu predobradu za razmatrani problem, ali su u nastavku prikazane određene karakteristike audio datoteke kako bi se dobio bolji uvid u podatke. U nastavku će biti prikazan oblik zvučnog signala, frekvencija signala, spektrogram te spektrogram mel koeficijenata.



*Slika 2 Valni oblik audio datoteke*

Na slici 2 vidljiv je valni oblik jedne audio datoteke iz skupa podataka. Najuočljivije promjene amplitude dešavaju se u trenutku izgovaranja riječi. Ispod je prikazan isječak koda kojim se dobio prikazani graf. Frekvencija uzorkovanja signala iznosi 16kHz.

```
# Waveform
audio_file_path = '/content/extracted_words/train/right/1004.wav'
signal, sr = librosa.load(audio_file_path, sr=16000)
librosa.display.waveshow(signal, sr=sr)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.show
```

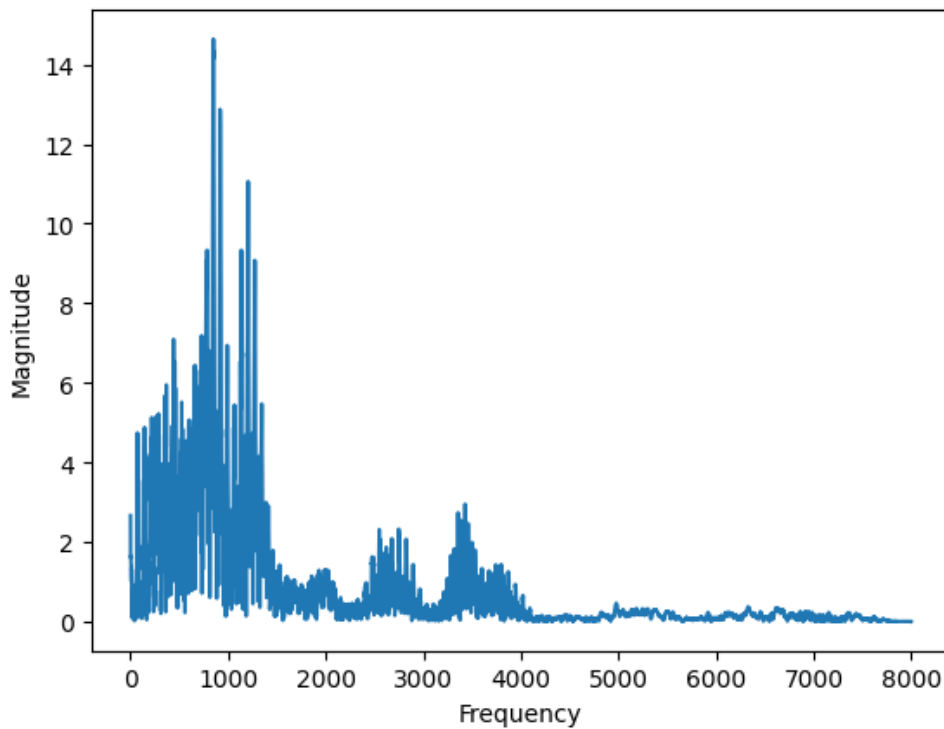
Da bi se prikazala promjena frekvencije audio signala kroz vrijeme, bilo je potrebno napraviti Fourier-ovu transformaciju signala što je prikazano isječkom koda u nastavku.

```
# FFT
fft = np.fft.fft(signal)
magnitude = np.abs(fft)
frequency = np.linspace(0, sr, len(magnitude))

left_frequency = frequency[:int(len(frequency)/2)]
left_magnitude = magnitude[:int(len(magnitude)/2)]
```

```
plt.plot(left_frequency, left_magnitude)
plt.xlabel('Frequency')
plt.ylabel('Magnitude')
plt.show
```

Prikazani odnos frekvencije i magnitude govori koliko pojedina frekvencija pridonosi cjelokupnom zvuku. To je vidljivo na slici 3. Prikazana je polovica frekvencije uzorkovanja s obzirom da je zbog svojstava Fourier-ove transformacije druga polovica simetrična prvoj te se ponavljaju iste informacije.



*Slika 3 Odnos frekvencije i magnitude signala*

Gornji prikaz je statičan, odnosno imamo utjecaj pojedinih frekvencija na cjelokupan zvuk u jednom trenutku.

Da bi dobili uvid u to kako pojedina frekvencija utječe na cjelokupan zvuk tijekom vremena, možemo iskoristiti spektrogram koji će nam dati informacije kako se amplituda ponaša u ovisnosti o frekvenciji i vremenu.

```
# STFT - spectrogram

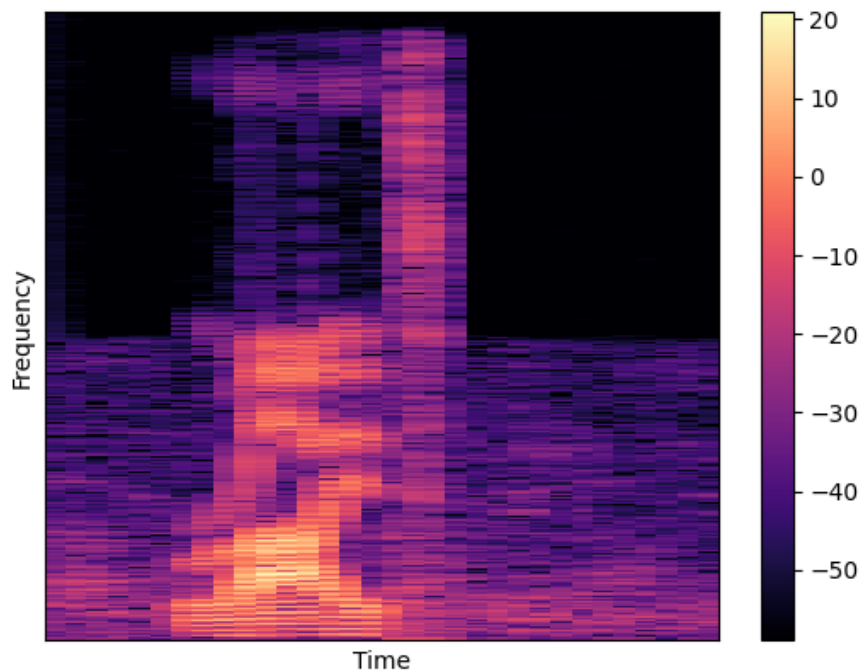
n_fft = 2048
hop_length = 512

stft = librosa.core.stft(signal, hop_length=hop_length, n_fft=n_fft)
spectrogram = np.abs(stft)
```

```
log_spectrogram = librosa.amplitude_to_db(spectrogram)

librosa.display.specshow(log_spectrogram, sr=sr, hop_length=hop_length)
plt.xlabel('Time')
plt.ylabel('Frequency')
plt.colorbar()
plt.show
```

Isječak koda iznad prikazuje kako se došlo do spektrograma, a na slici ispod prikazan je i sam spektrogram. Najglasniji zvukovi prikazani su svjetlijim bojama.

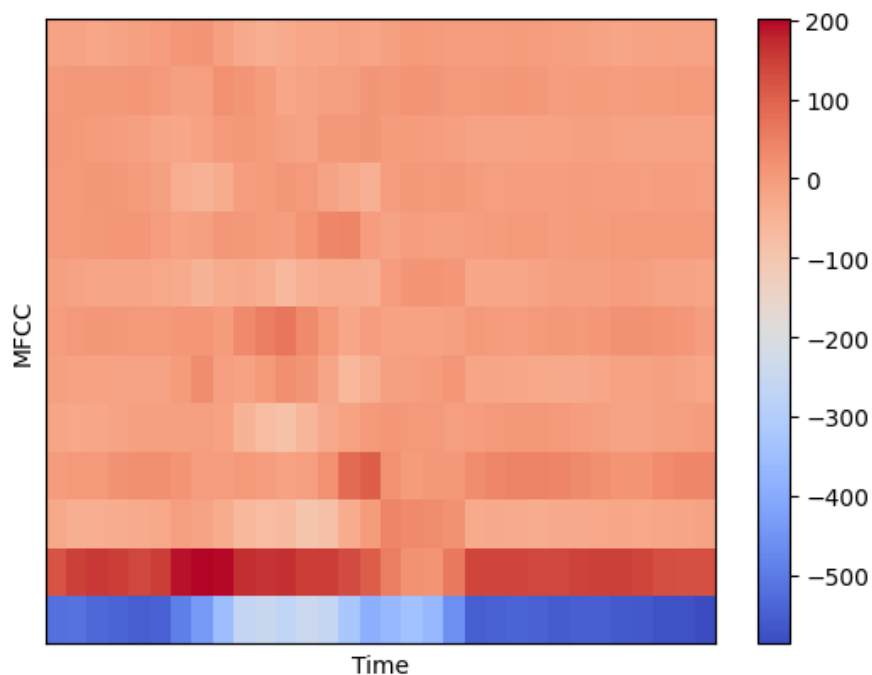


*Slika 4 Spektrogram audio signala*

Slika 5 prikazuje mel spektrogram koji analizira frekvenciju temeljenu na mel ljestvici koja je opazajno bliža načinu na koji ljudi doživljavaju zvuk. Ispod je prikazan isječak koda pomoću kojeg se dobio prikazani spektrogram.

```
# MFCCs

mffcs = librosa.feature.mfcc(y=signal, sr=sr, n_mfcc=13)
librosa.display.specshow(mffcs, sr=sr, hop_length=hop_length)
plt.xlabel('Time')
plt.ylabel('MFCC')
plt.colorbar()
plt.show
```



Slika 5 Mel spektrogram

Ukupan broj riječi u novonastalom podatkovnom skupu je 14 254. Od toga 20% riječi nalazi se u validacijskom i testnom skupu (po 10% u svakom), a preostalih 80% u trening skupu. Tablica 1 prikazuje podjelu riječi, to jest zvučnih datoteka, u tri skupa.

Tablica 1 Broj riječi po skupovima

Train	Validation	Test
11 403	1 426	1 425

Prikazani podatkovni skup korišten je za učenje modela. Iz podatkovnog skupa u kojemu su na svaku riječ dodane smetnje također je izvučeno spomenutih 10 riječi i na tom skupu je također testiran rad modela, a rezultati će biti kasnije spomenuti.

## Opis korištenih metoda/algoritama

Za rješenje zadatka kreiran je model konvolucijske neuronske mreže prikazan na slici 6.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 129, 71, 512)	5120
batch_normalization (Batch Normalization)	(None, 129, 71, 512)	2048
max_pooling2d (MaxPooling2D)	(None, 64, 35, 512)	0
conv2d_1 (Conv2D)	(None, 64, 35, 256)	1179904
batch_normalization_1 (Batch Normalization)	(None, 64, 35, 256)	1024
max_pooling2d_1 (MaxPooling2D)	(None, 32, 17, 256)	0
dropout (Dropout)	(None, 32, 17, 256)	0
conv2d_2 (Conv2D)	(None, 32, 17, 128)	295040
batch_normalization_2 (Batch Normalization)	(None, 32, 17, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 16, 8, 128)	0
flatten (Flatten)	(None, 16384)	0
dense (Dense)	(None, 512)	8389120
batch_normalization_3 (Batch Normalization)	(None, 512)	2048
dense_1 (Dense)	(None, 256)	131328
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 10)	2570
Total params: 10008714 (38.18 MB)		
Trainable params: 10005898 (38.17 MB)		
Non-trainable params: 2816 (11.00 KB)		

Slika 6 Izgled modela konvolucijske neuronske mreže

Model se sastoji od tri konvolucijska sloja nakon kojih slijedi batch normalizacija (engl. *BatchNormalization*) popraćena slojem sažimanja po maksimalnoj vrijednosti (engl. *Max Pooling*). Nakon drugog sloja sažimanja po maksimalnoj vrijednosti dodan je sloj s nasumičnim izbacivanjem neurona (engl. *Dropout*) s vrijednošću 0.25. Mreža ima i tri potpuno povezana sloja gdje nakon drugog ponovno slijedi sloj s nasumičnim izbacivanjem neurona.

Svi konvolucijski slojevi popraćeni su *ReLU* aktivacijskom funkcijom, kao i dva potpuno povezana sloja dok je zadnji potpuno povezani sloj popraćen *Softmax* aktivacijskom funkcijom kako bi se na izlazu dobila vrijednost između 0 i 1.

Gubitak koji je korišten pri treniranju modela je kategorički gubitak unakrsne entropije (engl. *Categorical CrossEntropy Loss*) koji se još naziva i *Softmax Loss* koji povezuje *Softmax* aktivacijsku funkciju i gubitak unakrsne entropije koji se koriste za višeklasnu klasifikaciju.



Korišten je *Adadelta* optimizator sa stopom učenja od 0.01 i regularizacijskim parametrom *weight\_decay* s iznosom od 0.00001 kako bi se ostvarila bolja generalizacija tijekom učenja.

Kako bi se izbjeglo pretjerano usklađivanje na podatke, dodani su ranije spomenuti slojevi *batch* normalizacije i sloj s nasumičnim izbacivanjem neurona. Također je implementirano i rano zaustavljanje (engl. *Early Stopping*) koje prati točnost klasifikacije na validacijskom skupu. Iz istog razloga je prilikom treniranja korišten i validacijski skup.

Model je treniran kroz 20 epoha s veličinom *batch-a* koja iznosi 32.

Postavljanje navedenih parametara za treniranje prikazano je isječkom koda ispod.

```
model.compile(loss=keras.losses.categorical_crossentropy,
               optimizer=keras.optimizers.Adadelta( learning_rate=0.01,
weight_decay=0.00001),
               metrics=['accuracy'])
model.build()
model.summary()

epochs = 20
batch_size = 32
callbacks = [
    EarlyStopping(
        monitor='val_accuracy',
        patience=4,
        mode='max',
        verbose=1)
]
```

Proces treniranja modela kroz svih 20 epoha prikazan je na slici 7. Također je vidljivo i kako su se iznos točnosti i gubitka mijenjali na trening i validacijskom skupu kroz epohe. Postignuta je točnost od 96% na trening skupu te točnost iznosa 95% na validacijskom skupu. Vidljivo je da iznos gubitka na oba skupa opada kako se mijenjaju epohe, a točnost raste iz čega je zaključeno kako nije došlo do pretjeranog usklađivanja na podatke

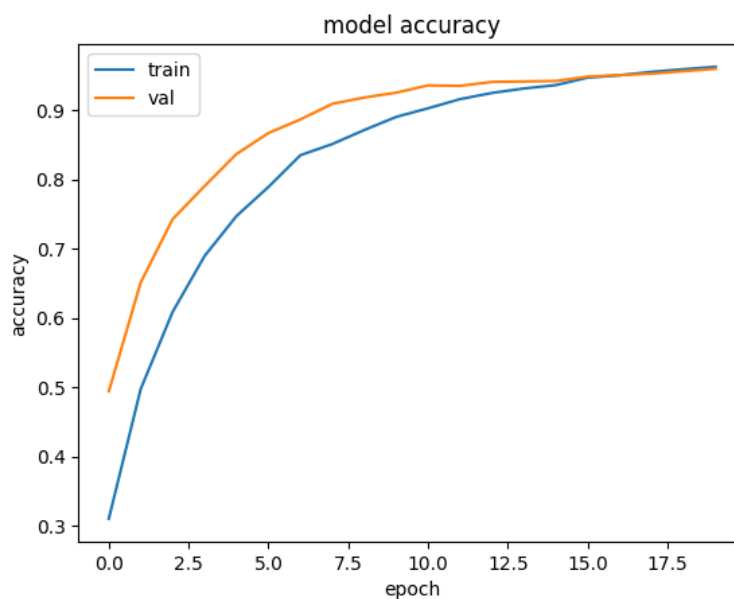
```

Epoch 1/20
357/357 [=====] - 82s 203ms/step - loss: 2.0442 - accuracy: 0.3103 - val_loss: 1.5293 - val_accuracy: 0.4940
Epoch 2/20
357/357 [=====] - 71s 200ms/step - loss: 1.4840 - accuracy: 0.4975 - val_loss: 1.2040 - val_accuracy: 0.6512
Epoch 3/20
357/357 [=====] - 72s 203ms/step - loss: 1.1756 - accuracy: 0.6087 - val_loss: 0.9424 - val_accuracy: 0.7425
Epoch 4/20
357/357 [=====] - 74s 207ms/step - loss: 0.9457 - accuracy: 0.6893 - val_loss: 0.7530 - val_accuracy: 0.7902
Epoch 5/20
357/357 [=====] - 73s 204ms/step - loss: 0.7783 - accuracy: 0.7470 - val_loss: 0.5997 - val_accuracy: 0.8365
Epoch 6/20
357/357 [=====] - 73s 205ms/step - loss: 0.6441 - accuracy: 0.7889 - val_loss: 0.5012 - val_accuracy: 0.8667
Epoch 7/20
357/357 [=====] - 74s 208ms/step - loss: 0.5389 - accuracy: 0.8347 - val_loss: 0.4222 - val_accuracy: 0.8863
Epoch 8/20
357/357 [=====] - 74s 206ms/step - loss: 0.4719 - accuracy: 0.8508 - val_loss: 0.3650 - val_accuracy: 0.9088
Epoch 9/20
357/357 [=====] - 75s 211ms/step - loss: 0.4065 - accuracy: 0.8710 - val_loss: 0.3170 - val_accuracy: 0.9179
Epoch 10/20
357/357 [=====] - 74s 207ms/step - loss: 0.3505 - accuracy: 0.8899 - val_loss: 0.2750 - val_accuracy: 0.9249
Epoch 11/20
357/357 [=====] - 73s 205ms/step - loss: 0.3101 - accuracy: 0.9024 - val_loss: 0.2509 - val_accuracy: 0.9354
Epoch 12/20
357/357 [=====] - 74s 208ms/step - loss: 0.2717 - accuracy: 0.9155 - val_loss: 0.2277 - val_accuracy: 0.9347
Epoch 13/20
357/357 [=====] - 74s 208ms/step - loss: 0.2459 - accuracy: 0.9245 - val_loss: 0.2102 - val_accuracy: 0.9404
Epoch 14/20
357/357 [=====] - 74s 208ms/step - loss: 0.2232 - accuracy: 0.9309 - val_loss: 0.1943 - val_accuracy: 0.9411
Epoch 15/20
357/357 [=====] - 74s 207ms/step - loss: 0.2032 - accuracy: 0.9358 - val_loss: 0.1794 - val_accuracy: 0.9418
Epoch 16/20
357/357 [=====] - 74s 207ms/step - loss: 0.1755 - accuracy: 0.9466 - val_loss: 0.1707 - val_accuracy: 0.9481
Epoch 17/20
357/357 [=====] - 74s 208ms/step - loss: 0.1661 - accuracy: 0.9500 - val_loss: 0.1659 - val_accuracy: 0.9502
Epoch 18/20
357/357 [=====] - 73s 205ms/step - loss: 0.1491 - accuracy: 0.9549 - val_loss: 0.1509 - val_accuracy: 0.9523
Epoch 19/20
357/357 [=====] - 74s 207ms/step - loss: 0.1352 - accuracy: 0.9587 - val_loss: 0.1453 - val_accuracy: 0.9558
Epoch 20/20
357/357 [=====] - 73s 204ms/step - loss: 0.1292 - accuracy: 0.9622 - val_loss: 0.1401 - val_accuracy: 0.9593

```

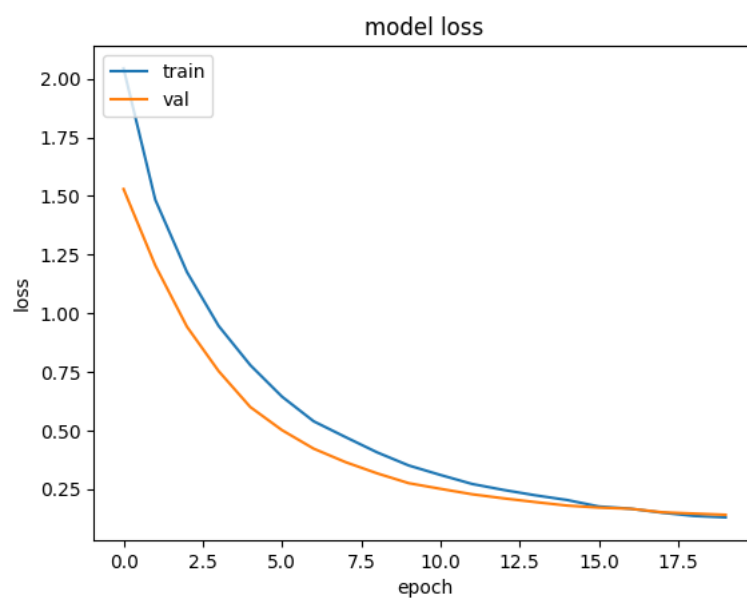
*Slika 7 Treniranje modela*

Na slici 8 nalazi se graf koji prikazuje promjenu točnosti na trening i validacijskom skupu kroz epohe.



*Slika 8 Točnost na trening i validacijskom skupu*

Slika 9 prikazuje kako se mijenjao gubitak na trening i validacijskom skupu kroz epohe.



*Slika 9 Gubitak na trening i validacijskom skupu*

## Evaluacija izgrađenog modela

Izgrađeni model je evaluiran na dva preuzeta testna skupa. U prvom se nalaze riječi bez izgovorene bez smetnji u vidu pozadinskih zvukova, a u drugom skupu nalaze se iste riječi, ali se slušanjem jasno mogu razaznati smetnje koje se pojavljuju u pozadini.

Također je prikupljen skup podataka u kojem se nalaze audio snimke riječi koje su izgovarale stvarne osobe. Svaku riječ izgovorilo je pet osoba po pet puta. Prikupljen skup podataka korišten je za procjenu rada modela.

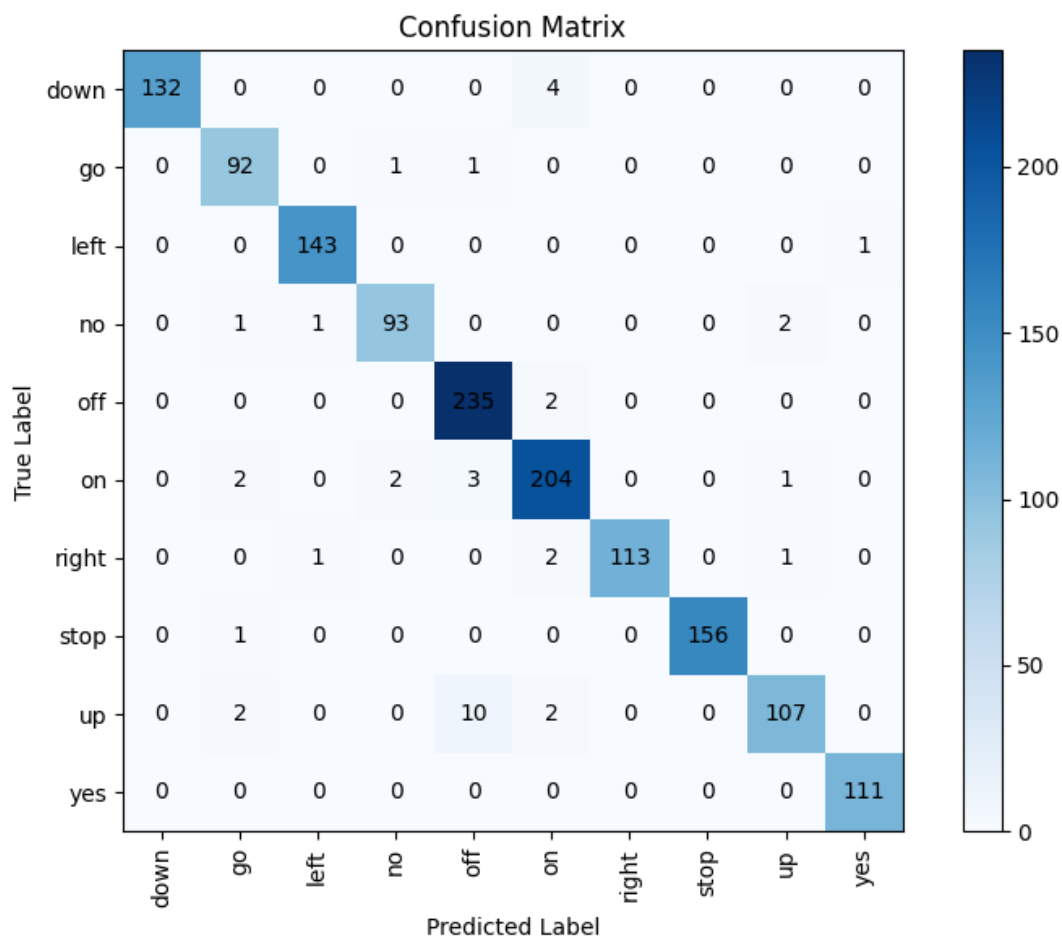
Rezultati dobiveni na testnom skupu bez smetnji koji se sastoji od 1 426 zvučnih datoteka prikazani su na slici ispod.

Model accuracy: 0.9712482468443198					
	precision	recall	f1-score	support	label
7	0.985401	0.992647	0.989011	136.0	stop
3	0.981308	0.990566	0.985915	106.0	no
9	0.992126	0.976744	0.984375	129.0	yes
0	0.982301	0.982301	0.982301	113.0	down
1	0.990196	0.971154	0.980583	104.0	go
2	0.973333	0.979866	0.976589	149.0	left
macro avg	0.973344	0.971310	0.972252	1426.0	macro avg
weighted avg	0.971429	0.971248	0.971273	1426.0	weighted avg
micro avg	0.971248	0.971248	0.971248	1426.0	micro avg
samples avg	0.971248	0.971248	0.971248	1426.0	samples avg
5	0.967593	0.972093	0.969838	215.0	on
6	0.982456	0.941176	0.961373	119.0	right
4	0.954357	0.966387	0.960334	238.0	off
8	0.924370	0.940171	0.932203	117.0	up

Slika 10 Performanse modela na prvom testnom skupu

Na prvom testnom skupu dobivena je točnost od 97% (u tablici na slici prikazano kao *micro avg*).

Na slici je za svaku pojedinu riječ prikazan iznos preciznosti (engl. *precision*), odziva (engl. *recall*) i F1 mjera (engl. *F1-score*). *Macro avg* predstavlja srednju vrijednost dobivenih F1 mjera. *Weighted avg* dobiven je kao srednja vrijednost F1 mjera za svaku pojedinu klasu pomnoženu s iznosom *support-a* za tu klasu.



Slika 11 Matrica zabune na testnim podacima

Slika 11 prikazuje matricu zabune dobivenu na testnom skupu podataka. Na glavnoj dijagonali nalaze se točne predikcije, a izvan dijagonale netočne. Također je vidljivo s kojim riječima klasifikator zamjenjuje pojedine riječi.

Na drugom testnom skupu koji se sastoji od 14 254 zvučnih datoteka sa pozadinskim smetnjama dobivena je točnost od 84% što je vidljivo na slici 12 kao i ostale metrike za pojedinu klasu.

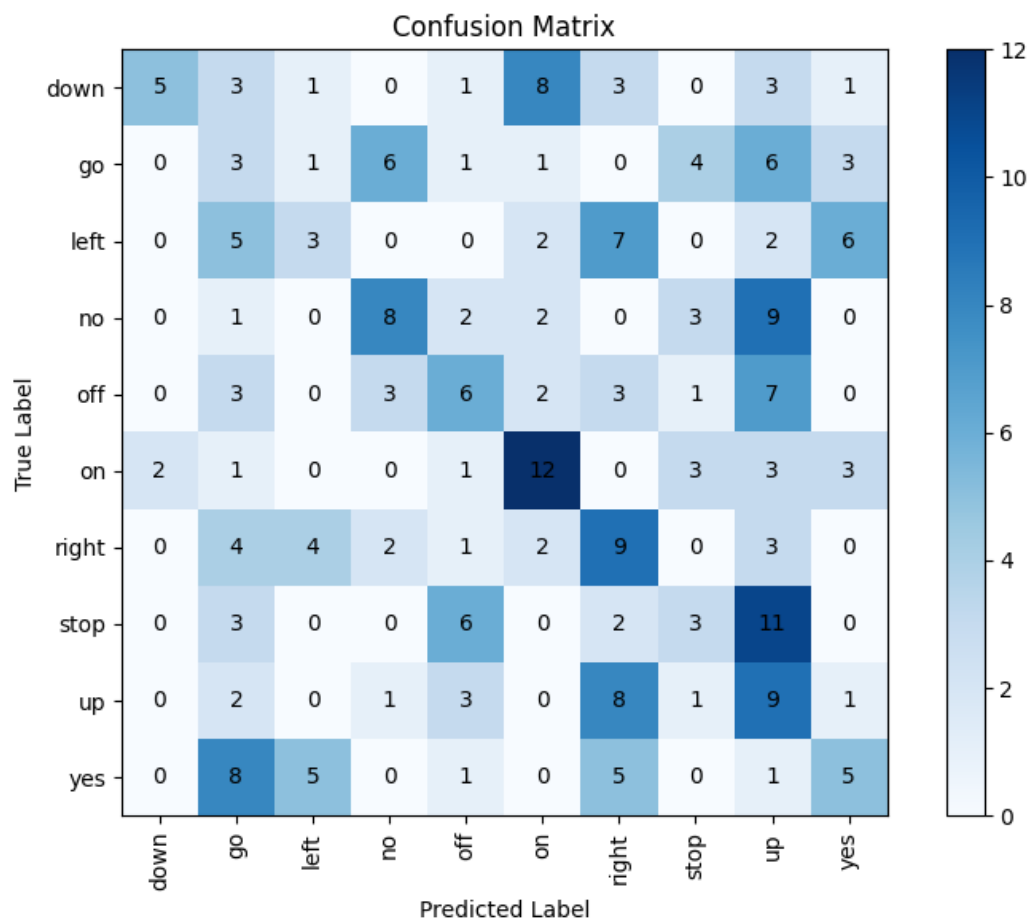
Model accuracy: 0.8469201627613302					
	precision	recall	f1-score	support	label
7	0.994894	0.918519	0.955182	1485.0	stop
9	0.960616	0.901929	0.930348	1244.0	yes
2	0.925571	0.845791	0.883885	1485.0	left
6	0.948864	0.785266	0.859348	1276.0	right
weighted avg	0.857265	0.846920	0.849252	14254.0	weighted avg
macro avg	0.864014	0.839530	0.848870	14254.0	macro avg
micro avg	0.846920	0.846920	0.846920	14254.0	micro avg
samples avg	0.846920	0.846920	0.846920	14254.0	samples avg
0	0.886810	0.797980	0.840053	1188.0	down
3	0.830737	0.835946	0.833333	957.0	no
4	0.779587	0.874777	0.824444	2244.0	off
5	0.784234	0.866248	0.823203	2228.0	on
1	0.872792	0.771875	0.819237	960.0	go
8	0.656033	0.796967	0.719665	1187.0	up

Slika 12 Performanse modela na drugom testnom skupu

Kako je već spomenuto, model je evaluiran i na skupu podataka koji se sastoji od snimljenih audio datoteka. Kako bi se olakšao rad sa snimljenim datotekama, napisana je skripta za snimanje audio datoteka. U skripti je zadana frekvencija uzorkovanja od 16kHz i trajanje svake snimke ograničeno je na 1s kako bi snimke odgovarale snimkama iz podatkovnog skupa na osnovu kojeg je model izgrađen.

Korištenjem napisane skripte snimljeno je 250 riječi.

Rezultati dobiveni na vlastitim podaci prikazani su matricom zabune i vrijednostima korištenih metrika na slikama ispod.



Slika 13 Matrica zabune na vlastitim podacima

Model accuracy: 0.252

	precision	recall	f1-score	support	label
5	0.413793	0.480	0.444444	25.0	on
3	0.400000	0.320	0.355556	25.0	no
0	0.714286	0.200	0.312500	25.0	down
6	0.243243	0.360	0.290323	25.0	right
4	0.272727	0.240	0.255319	25.0	off
weighted avg	0.297907	0.252	0.252056	250.0	weighted avg
macro avg	0.297907	0.252	0.252056	250.0	macro avg
micro avg	0.252000	0.252	0.252000	250.0	micro avg
samples avg	0.252000	0.252	0.252000	250.0	samples avg
8	0.166667	0.360	0.227848	25.0	up
9	0.263158	0.200	0.227273	25.0	yes
2	0.214286	0.120	0.153846	25.0	left
7	0.200000	0.120	0.150000	25.0	stop
1	0.090909	0.120	0.103448	25.0	go

Slika 14 Performanse modela na vlastitim podacima

Točnost modela na prikupljenom skupu podataka iznosi 25%. Ako se usporedi iznos točnosti od 97% koji je ostvaren korištenjem testnog skupa koji se sastoji od podataka iz početnog podatkovnog skupa, rezultat nije zadovoljavajuć. Međutim, dobiveni rezultat je očekivan s obzirom da se u prvom testnom skupu nalaze sintetički, a sam model je treniran na sintetičkim podacima koje „izgovara“ računalni program. Na matrici zabune vidljivo je koje riječi je model zamijenio što je vjerojatno posljedica nejasnoće izgovora riječi od strane stvarne osobe.



## Demonstracija rada

Kako je već spomenuto, kao pomoć pri snimanju zvučnih datoteka napisana je skripta *app\_for\_recording.py*. Kod skripte prikazan je u nastavku.

```
import streamlit as st
import numpy as np
import sounddevice as sd
from scipy.io.wavfile import write

# Skripta pruža mogućnost snimanja 30 zvučnih datoteka koje sprema u mapu u koju
# je spremljena i sama skripta
sample_rate = 16000

def record_audio(file_number):
    duration = 1

    with st.spinner("Recording audio {file_number}..."):
        audio_data = sd.rec(int(duration * 16000), samplerate=16000, channels=1,
dtype='int16')
        sd.wait()
        st.success("Recording finished!")

    file_path = f'recorded_audio_{file_number}.wav'
    write(file_path, sample_rate, audio_data.flatten())

    return audio_data

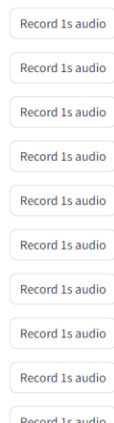
def main():
    audio_data = None
    st.title("Klasifikacija zvučnih naredbi")

    for i in range(1, 31):
        if st.button("Record 1s audio", key=f"record_button_{i}"):
            audio_data = record_audio(i)

if __name__ == "__main__":
    main()
```

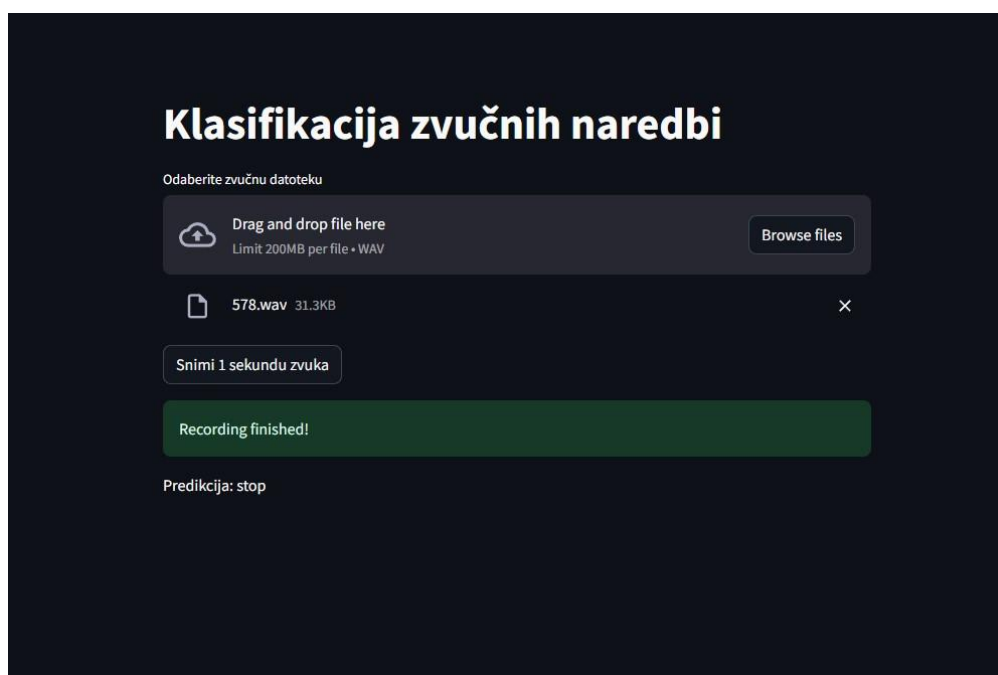
Skripta se pokreće ukucavanjem naredbe „*streamlit run app\_for\_recording.py*“ u terminal nakon čega se otvara aplikacija za snimanje u web pregledniku. Izgled aplikacije za snimanje prikazan je na slici u nastavku.

## Klasifikacija zvučnih naredbi



Slika 15 Izgled skripte za snimanje audio datoteka

Nadalje, kreirana je *Streamlit* aplikacija koja omogućava učitavanje ili snimanje vlastite audio datoteke kako bi se dobila predikcija modela. Izgled aplikacije i kod prikazan je slikama ispod.



Slika 16 Izgled streamlit\_app aplikacije

Skripta omogućava učitavanje proizvoljne .wav datoteke za klasifikaciju zvuka, moguće je učitati više datoteka ali ne odjednom. Također je omogućeno snimanje zvuka preko mikrofona te klasifikacija tog zvuka pomoću učitano modela. U nastavku se vidi kod kojim je napravljena pomenuta aplikacija.

```

#definiranje klase
y = ['down', 'go', 'left', 'no', 'off', 'on', 'right', 'stop', 'up', 'yes']
mlb = MultilabelBinarizer()
mlb.fit(pd.Series(y).fillna("missing").str.split(' '))
y_mlb = mlb.transform(pd.Series(y).fillna("missing").str.split(' '))

#funkcija za klasificiranje snimljenog zvuka
def classify_sound_record(samples, sample_rate):

    frequencies, times, spectrogram = signal.spectrogram(samples, sample_rate)
    new_X = spectrogram.reshape((1, spectrogram.shape[0], spectrogram.shape[1], 1))

    prediction = model.predict(new_X)
    predicted_class = np.argmax(prediction)
    predicted_word = mlb.classes_[predicted_class]
    return predicted_word

#funkcija za klasificiranje učitano zvuka
def classify_sound_read(file_path):

    sample_rate, samples = wavfile.read(file_path)
    frequencies, times, spectrogram = signal.spectrogram(samples, sample_rate)
    new_X = spectrogram.reshape((1, spectrogram.shape[0], spectrogram.shape[1], 1))

    prediction = model.predict(new_X)
    predicted_class = np.argmax(prediction)
    predicted_word = mlb.classes_[predicted_class]
    return predicted_word

#funkcija za snimanje zvuka
def record_audio():
    duration = 1

    with st.spinner("Recording..."):
        audio_data = sd.rec(int(duration * 16000), samplerate=16000, channels=1, dtype='int16')
        sd.wait()
    st.success("Recording finished!")
    return audio_data

def main():
    audio_data = None
    st.title("Klasifikacija zvučnih naredbi")

    audio_file = st.file_uploader("Odaberite zvučnu datoteku", type=["wav"])

    #ako je stisnuto tipkalo snjimi zvuk i klasificiraj ga
    if st.button("Snimi 1 sekundu zvuka"):
        audio_data = record_audio()
        if audio_data is not None:
            prediction = classify_sound_record(audio_data.flatten(), sample_rate)
            st.write(f"Predikcija: {prediction}")
            prediction = None

    #ako je učitano zvuk klasificiraj ga
    elif audio_file is not None:
        prediction = classify_sound_read(audio_file)
        st.write(f"Predikcija: {prediction}")
        prediction = None

```

Imamo tri funkcije, prva je *classify\_sound\_record* koja služi za klasifikaciju snimljene zvučne datoteke. Funkcija *classify\_sound\_read* klasificira učitano zvučnu datoteku, te funkcija *record\_audio* služi za snimanje zvuka.

## **Zaključak**

U ovom izvješću prikazan je kreirani model sustava za klasifikaciju glasovnih naredbi. Ideja je bila kreirati rješenje koje bi se moglo koristiti na sličan način kao neki od sustava koji prime određenu riječ, prepoznaju je i iskoriste kao naredbu na osnovu koje izvrše određenu radnju.

Za realizaciju rješenja korištena je konvolucijska neuronska mreža s kojom je postignuta točnost od 97% na podacima bez smetnji, odnosno 84% na podacima gdje su prisutne i smetnje. Na skupu podataka koji se sastoji od riječi koje su izgovarale stvarne osobe dobivena je točnost od 25%.

Poboljšanje rezultata predikcija modela na vlastitim podacima zahtijeva daljnje poboljšanje samog modela što bi se moglo pokušati ostvariti korištenjem većeg skupa podataka koji bi se sastojao i od stvarnih podataka koji nisu nastali korištenjem računalnih programa, mijenjanjem parametara treniranja ili izgradnjom složenijeg modela.

## Literatura

- [1] Sharma, Mridusmita & Sarma, Kandarpa. (2015). Soft-Computational Techniques and Spectro-Temporal Features for Telephonic Speech Recognition: An Overview and Review of Current State of the Art. 10.4018/978-1-4666-9474-3.ch006.
- [2] Buchner J. Synthetic Speech Commands: A public dataset for single-word speech recognition, 2017. Available from <https://www.kaggle.com/jbuchner/synthetic-speech-commands-dataset>