

Lecture 12

Speech Recognition

cscie-89 Deep Learning

- [Learning Objectives](#)
 - [Sound into numbers](#)
- [Visualization of the recordings - input features](#)
 - [Wave and spectrogram](#)
 - [MFCC](#)
 - [Silence removal](#)
 - [Resampling - dimensionality reductions](#)
 - [Features extraction steps](#)
- [Dataset investigation](#)
 - [Number of files](#)
 - [Mean spectrograms and fft](#)
 - [Deeper into recordings](#)
 - [Length of recordings](#)
 - [Note on Gaussian Mixtures modeling](#)
 - [Frequency components across the words](#)
 - [Anomaly detection](#)
- [Where to look for inspiration](#)

Learning Objectives

Towards the end of the lecture we will build a Simple Speech Recognition Engine using Convolutional Layers in Keras which will classified different sounds.

- We will learn how to perform feature extraction (MFCC) and train the model
- We will demonstrate that our model can recognize words.
- Determine different words captured by the microphone (Test)

We learned about CNNs, RNNs and classified MNIST for images dataset using Deep Learning libraries. We learned how to take an image and treat it as an array of numbers so that we can feed directly into a neural network for image recognition and now we can do the same with speech recognition audio files!

Sound is transmitted as waves. How do we turn sound waves into numbers?

Speech recognition is complex and tricky so to keep everything simple we will start with a practical tutorial using word recognition and Kaggle datasets of number words.

Data and Code

Data used in the examples in this notebook are downloaded from

https://storage.cloud.google.com/download.tensorflow.org/data/speech_commands_v0.02.tar.gz

(https://storage.cloud.google.com/download.tensorflow.org/data/speech_commands_v0.02.tar.gz) site. The directory into which we expanded collections of single spoken words: bird, dog, cat, four, happy,..., etc, we called data .

To reduce the processing time, we trained the neural network using a subset of data. You can do as you please. You can work with small or the entire data set.

Static images in this notebook should resided in directory named images . The content of the directory images is provide on the class site, week 12 folder.

Code used in the examples below is borrowed from: <https://blog.manash.me/building-a-dead-simple-word-recognition-engine-using-convnet-in-keras-25e72c19c12b> (<https://blog.manash.me/building-a-dead-simple-word-recognition-engine-using-convnet-in-keras-25e72c19c12b>)

Speech recordings are broken up into little chunks of phonetic signals of typical duration of 25 msec. Those windows are overlapping and slide (typically) 10 msec at a time.

The distinguishable units of speech are called phoneme.


Phoneme units are grouped into words

Spoken words are a sequence of sound waves

We will create vectors from the audio signal in order to feed our deep learning model. We work with RNN (LSTM)s to processes sequences.

In the follwing examples we will actually use convolutional neural networks for the classification of speech samples.

Inputs will be tensors of features representing speech utterances.

 train-words.png

We will perform classification using audio files with small words such as "seven", "happy", "yes", others

How to embed the audio into vector space?

There are many techniques and some Python packages solely for extraction of audio features.

MFCC encoding (Mel Frequency Cepstral Coefficients) is a very effective technique for encoding of speech signals. That technique was used for classica non-nural network speech recognition.

Various libraries we might or might not use

```
In [1]: import os
from os.path import isdir, join
from pathlib import Path
import pandas as pd

# Math
import numpy as np
from scipy.fftpack import fft
from scipy import signal
from scipy.io import wavfile
import librosa

from sklearn.decomposition import PCA

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
import IPython.display as ipd
import librosa.display

import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
import pandas as pd

%matplotlib inline
```

C:\ProgramData\Anaconda3\lib\site-packages\librosa\util\decorators.py:9: NumbaDeprecationWarning: An import was requested from a module that has moved location.

Import requested from: 'numba.decorators', please update to use 'numba.core.decorators' or pin to Numba version 0.48.0. This alias will not be present in Numba version 0.50.0.

```
from numba.decorators import jit as optional_jit
```

C:\ProgramData\Anaconda3\lib\site-packages\librosa\util\decorators.py:9: NumbaDeprecationWarning: An import was requested from a module that has moved location.

Import of 'jit' requested from: 'numba.decorators', please update to use 'numba.core.decorators' or pin to Numba version 0.48.0. This alias will not be present in Numba version 0.50.0.

```
from numba.decorators import jit as optional_jit
```

Audio Processing - Turning Sounds into Bits

The first step in any automatic speech recognition system is to extract features i.e. identify the components of the audio signal that are good for identifying the linguistic content and discarding all the other stuff which carries information like background noise, emotion etc.

The main point to understand about speech is that the sounds generated by a human are filtered by the shape of the vocal tract including tongue, teeth etc. This shape determines what sound comes out. If we can determine the shape accurately, this should give us an accurate representation of the phoneme being produced. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum, and the job of MFCCs is to accurately represent this envelope.

Speech Signals - Sound Waves & Digital Sampling

Think of Sound waves as one-dimensional. At every moment in time, they have a single value based on the height of the wave. To turn this sound wave into numbers, we just record of the height of the wave at equally-spaced points. This is called sampling.



We are taking a reading thousands of times a second and recording a number representing the height of the sound wave at that point in time. That's basically all an uncompressed .wav audio file is. "CD Quality" audio is sampled at 44.1khz (44,100 readings per second). But for speech recognition, a sampling rate of 16khz (16,000 samples per second) is enough to cover the frequency range of human speech.

Pre-processing our Sampled Sound Data and what is a MFCC?

From Niquist theorem's we know that we could perfectly reconstruct the original sound wave from the sample readings—as long as we sample at least twice as fast as the highest frequency we want to record.

Nearly everyone gets this wrong and assumes that using higher sampling rates always leads to better audio quality. It doesn't.

The most commonly used feature extraction method in automatic speech recognition (ASR) is Mel-Frequency Cepstral Coefficients (MFCC).

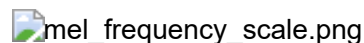
MFCC can create useful vectors from audio signal for and could server as numerical encoding of sound waves we need for deep learning models to work!

Mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear Mel scale of frequency.

Mel-frequency scale is described by the following formula expressing mels over frequency :

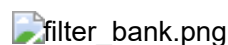


The mel scale, introduced by Stevens, Volkman, and Newman in 1937, is a perceptual scale of pitches judged by listeners to be equal in distance from one another. The reference point between this scale and normal frequency measurement is defined by assigning a perceptual pitch of 1000 mels to a 1000 Hz tone, 40 dB above the listener's threshold. Above about 500 Hz, increasingly large intervals are judged by listeners to produce equal pitch increments. As a result, four octaves on the hertz scale above 500 Hz are judged to comprise about two octaves on the mel scale. The name mel comes from the word melody to indicate that the scale is based on pitch comparisons.



Based on me frequency scale we conclude that in order to capture speech it is necessary to take many samples at low frequencies but fewer and fewer at higher frequencies.

This reflected in the design of the banks of filters which are used for speech recording. Typical recording software (hardware) operates as if it captures sounds through a set of filters with progressively larger and larger band-width.



MFCC vectors are feature vector containing "all" information about the sound waves representing human words. MFCC mimics some parts of the human speech production and speech perception. MFCC mimics the logarithmic perception of loudness and pitch of human auditory system and tries to eliminate speaker dependent characteristics by excluding the fundamental frequency and their harmonics. To represent the dynamic nature of speech the MFCC also includes the change of the feature vector over time as part of the feature vector.

Visualizing Sound

There are two theories of a human hearing - Place Theory ([https://en.wikipedia.org/wiki/Place_theory_\(hearing\)](https://en.wikipedia.org/wiki/Place_theory_(hearing))) (frequency-based) and Temporal theory ([https://en.wikipedia.org/wiki/Temporal_theory_\(hearing\)](https://en.wikipedia.org/wiki/Temporal_theory_(hearing))) In speech recognition, some people use as input spectrogram (<https://en.wikipedia.org/wiki/Spectrogram>) (frequencies). Others use more sophisticated features MFCC - Mel-Frequency Cepstral Coefficients. We rarely work with raw, temporal data.

Let's visualize some recordings!

Wave and spectrogram:

Choose and read some file:

```
In [2]: audio_path = 'data/'
filename = 'yes/0a7c2a8d_nohash_0.wav'
#filename = '/seven/6c968bd9_nohash_0.wav'
# sample_rate, samples = wavfile.read(str(audio_path) + filename)
samples, sample_rate = librosa.load(str(audio_path)+filename)
```

```
In [3]: print(samples.shape)
print("sample rate: ", sample_rate)

(22050,)
sample rate: 22050
```

Function that calculates spectrogram, log_specgram()

Note, that we are taking logarithm of spectrogram values. It will make our plot much more clear, moreover, it is strictly connected to the way people hear. We need to assure that there are no 0 values as input to logarithm.

```
In [4]: #def log_specgram(audio, sample_rate, window_size=20,
#               step_size=10, eps=1e-10):
def log_specgram(audio, sample_rate=16000, window_size=20,
                 step_size=10, eps=1e-10):
    if audio.ndim > 1 : #ignore channels 2+
        audio = audio[:,0]
    nperseg = int(round(window_size * sample_rate / 1e3))
    noverlap = int(round(step_size * sample_rate / 1e3))
    freqs, times, spec = signal.spectrogram(audio,
                                             fs=sample_rate,
                                             window='hann',
                                             nperseg=nperseg,
                                             noverlap=noverlap,
                                             detrend=False)
    return freqs, times, np.log(spec.T.astype(np.float32) + eps)
#return freqs, times, np.log(spec.T.astype(np.float32) + eps)
```

Frequencies are in the range (0, 8000). According to [Nyquist theorem \(https://en.wikipedia.org/wiki/Nyquist_rate\)](https://en.wikipedia.org/wiki/Nyquist_rate) we could faithfully reproduce speech signal from digital file created with sampling rate of 16,000/sec

Let's plot some spectrograms:

```

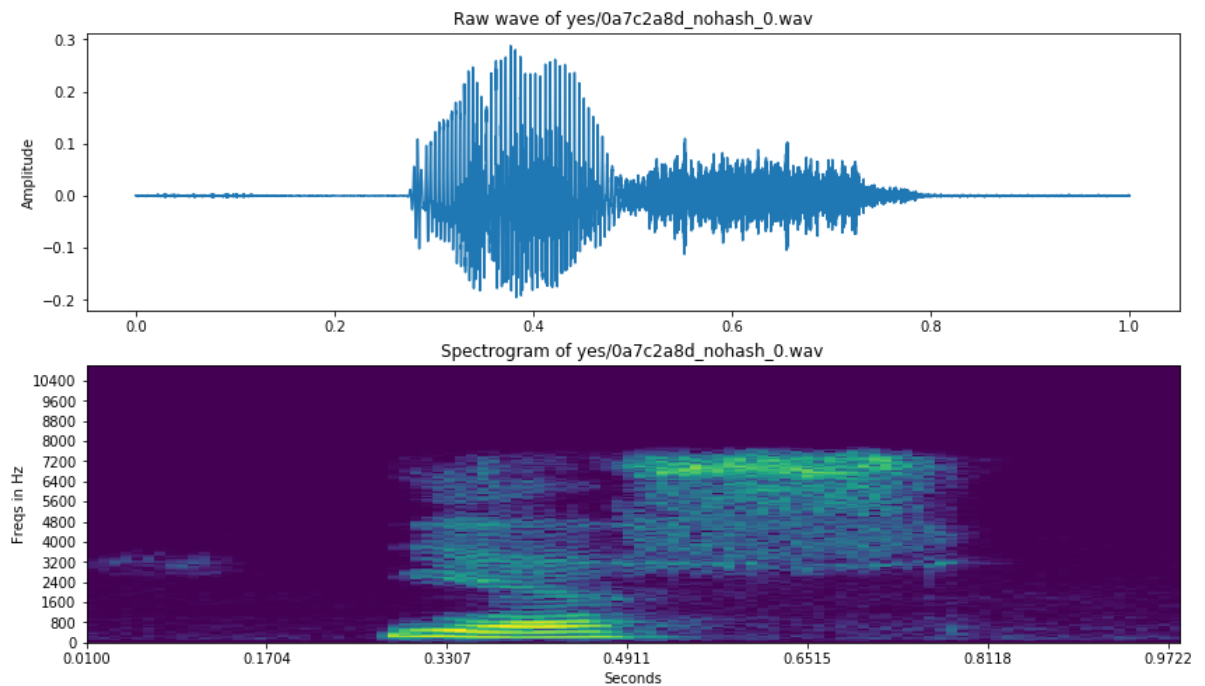
In [5]: freqs, times, spectrogram = log_specgram(samples, sample_rate)

fig = plt.figure(figsize=(14, 8))
ax1 = fig.add_subplot(211)
ax1.set_title('Raw wave of ' + filename)
ax1.set_ylabel('Amplitude')
ax1.plot(np.linspace(0, sample_rate/len(samples), sample_rate), samples)

ax2 = fig.add_subplot(212)
ax2.imshow(spectrogram.T, aspect='auto', origin='lower',
           extent=[times.min(), times.max(), freqs.min(), freqs.max()])
ax2.set_yticks(freqs[::16])
ax2.set_xticks(times[::16])
ax2.set_title('Spectrogram of ' + filename)
ax2.set_ylabel('Freqs in Hz')
ax2.set_xlabel('Seconds')

```

Out[5]: Text(0.5, 0, 'Seconds')



If we use spectrogram as an input features for Neural Network, we have to remember to normalize features. We would need to normalize all spectra in the dataset. We would use functions in the following cell, but not just for one spectrum, but rather all.

```

In [ ]: # Do not run this code
mean = np.mean(spectrogram, axis=0)
std = np.std(spectrogram, axis=0)
spectrogram = (spectrogram - mean) / std

```


There is an interesting fact to point out.

We have ~160 features for each frame, frequencies are between 0 and 8000. It means, that one feature corresponds to 50 Hz. However, frequency resolution of the ear is 3.6 Hz within the octave of 1000 – 2000 Hz. It means, that people are far more precise and can hear much smaller details than those represented by spectrograms like above.

Librosa, Python Sound Processing Library

It appears that `librosa` is the most popular sound processing library. To have Librosa work properly on Windows, you need to have package `ffmpeg` installed and its `ffmpeg/bin` directory placed in the `PATH` environmental variable. Install librosa version 0.6.3. At least this code has some issues with the newest version of that library.

```
In [ ]: # To load audio files and resample a signal at 16000Hz, you would use command
        # like:
        #librosa.load(audio_path, sr=16000)
        # To load files without resampling, do:
        #librosa.load(audio_path, sr=None)
```

Fourier Transformation & Mel-Frequency Spectrum

Fourier Transform:



The first processing step is the computation of the frequency domain representation of the input signal. This is achieved by computing the **Discrete Fourier Transform**.

Where N is the number of sampling points within a speech frame and the time frame τ . For implementations the Fast Fourier Transform, which is a variation of the Discrete Fourier Transformation optimized for speed, is used.

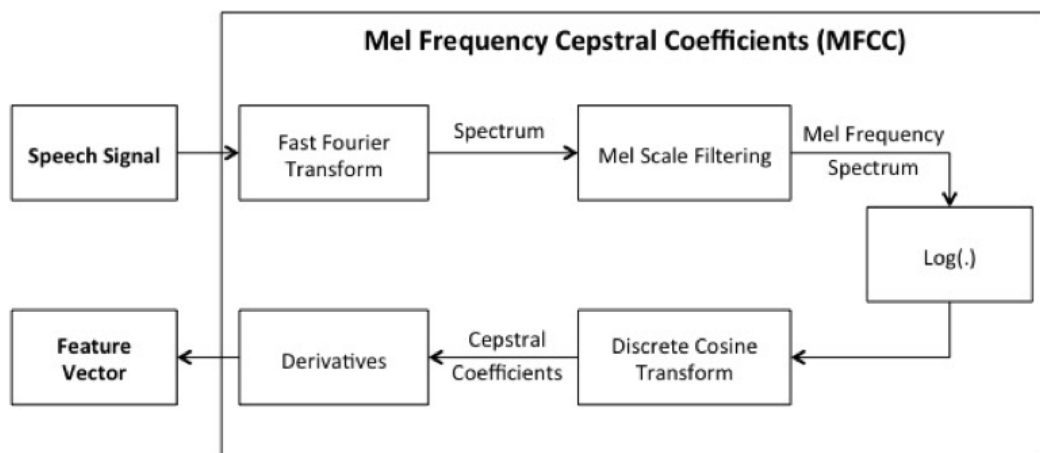
Mel-Frequency Spectrum (MFC):

The second processing step is the computation of the mel-frequency spectrum.

This processing step can be described by the amplitude of the band-pass filter with the index j at the frequency k and the power of each frequency band is computed with N_d different "band-pass filters". This filtering mimics the human ear because the **human auditory system uses the power over a frequency band as signal for further processing**.

```
In [6]: from IPython.display import Image
Image("images/MFCC_Flowchart.png")
```

Out[6]:



Resampled Sound

We now have an array of numbers with each number representing the sound wave's amplitude at 1/16,000th of a second intervals.

We could feed these numbers right into a neural network. But trying to recognize speech patterns by processing these samples directly is difficult. Instead, we can make the problem easier by doing some pre-processing on the audio data.

Let's start by grouping our sampled audio into 20-millisecond-long chunks. Here's our first 20 milliseconds of audio (i.e., our first 400 samples): Plotting those numbers as a simple line graph gives us a rough approximation of the original sound wave for that 20 millisecond period of time. This recording is only 1/50th of a second long. But even this short recording is a complex mish-mash of different frequencies of sound. There's some low sounds, some mid-range sounds, and even some high-pitched sounds sprinkled in. But taken all together, these different frequencies mix together to make up the complex sound of human speech.

To make this data easier for a neural network to process, we are going to break apart this complex sound wave into its component parts. We'll break out the low-pitched parts, the next-lowest-pitched-parts, and so on. Then by adding up how much energy is in each of those frequency bands (from low to high), we create a fingerprint of sorts for this audio snippet.

We do this using a mathematic operation called a **Fourier transform**. It breaks apart the complex sound wave into the simple sound waves that make it up. Once we have those individual sound waves, we add up how much energy is contained in each one.

The end result is a score of how important each frequency range is, from low pitch (i.e. bass notes) to high pitch. Each number below represents how much energy was in each 50hz band of our 20 millisecond audio clip. But this is a lot easier to see when you draw this as a chart. If we repeat this process on every 20 millisecond chunk of audio, we end up with a spectrogram (each column from left-to-right is one 20ms chunk).

Mel-Frequency Cepstrum Coefficients

Features we will extract from sound waves are Mel-Frequency Cepstrum Coefficients (MFCC)

```
In [7]: audio_path = './data/seven/6c968bd9_nohash_0.wav'
y, sr = librosa.load(audio_path)
# Let's make and display a mel-scaled power (energy-squared) spectrogram
S = librosa.feature.melspectrogram(y, sr=sr, n_mels=128)
# Convert to log scale (dB). We'll use the peak power (max) as reference.
log_S = librosa.power_to_db(S, ref=np.max)

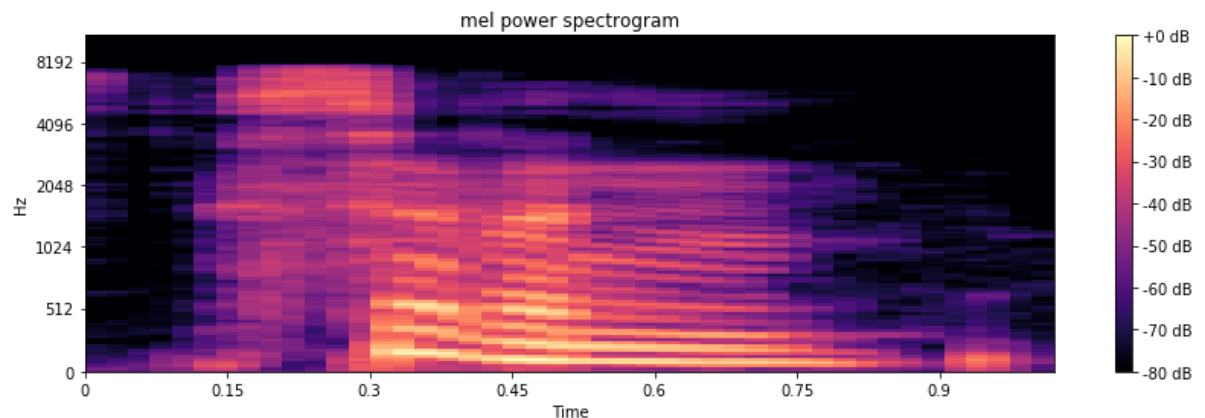
# Make a new figure
plt.figure(figsize=(12,4))

# Display the spectrogram on a mel scale
# sample rate and hop length parameters are used to render the time axis
librosa.display.specshow(log_S, sr=sr, x_axis='time', y_axis='mel')

# Put a descriptive title on the plot
plt.title('mel power spectrogram')

# draw a color bar
plt.colorbar(format='%+02.0f dB')

# Make the figure layout compact
plt.tight_layout()
```



```

In [8]: # Next, we'll extract the top 13 Mel-frequency cepstral coefficients (MFCCs)
mfcc      = librosa.feature.mfcc(S=log_S, n_mfcc=13)

# Let's pad on the first and second deltas while we're at it
delta_mfcc = librosa.feature.delta(mfcc)
delta2_mfcc = librosa.feature.delta(mfcc, order=2)

# How do they look? We'll show each in its own subplot
plt.figure(figsize=(12, 6))

plt.subplot(3,1,1)
librosa.display.specshow(mfcc)
plt.ylabel('MFCC')
plt.colorbar()

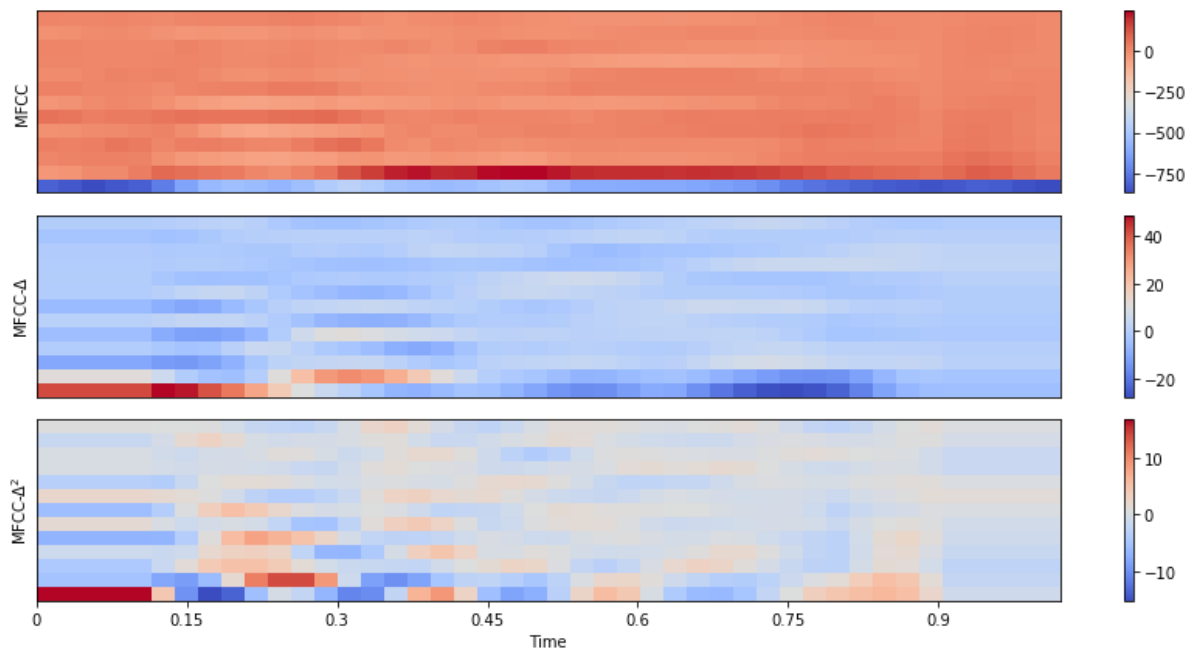
plt.subplot(3,1,2)
librosa.display.specshow(delta_mfcc)
plt.ylabel('MFCC- $\Delta$ ')
plt.colorbar()

plt.subplot(3,1,3)
librosa.display.specshow(delta2_mfcc, sr=sr, x_axis='time')
plt.ylabel('MFCC- $\Delta^2$ ')
plt.colorbar()

plt.tight_layout()

# For future use, we'll stack these together into one matrix
M = np.vstack([mfcc, delta_mfcc, delta2_mfcc])

```



In classical, but still important speech recognition systems, as well as in Deep Learning Speech Recognition systems, *MFCC* or similar features are taken as the input to the speech recognition systems instead of spectrograms.

However, in end-to-end (neural-network based) systems, the most common input features are probably raw spectrograms, or mel power spectrograms. For example *MFCC* decorrelates features, but NNs deal with correlated features well. Also, if you understand mel filters, you may consider their sensible usage.

Silence removal

Let's listen to that `yes` file

```
In [9]: ipd.Audio(samples, rate=sample_rate)
```

Out[9]:
0:00 / 0:01

Some VAD (Voice Activity Detection) will be really useful here. Although the words are short, there is a lot of silence in them. A decent VAD could reduce training size a lot, accelerating training speed significantly. Let's cut a bit of the file from the beginning and from the end and listen to it again (based on a plot above, we take samples from 4000 to 9600):

```
In [10]: samples_cut = samples[4000:9600]  
         ipd.Audio(samples_cut, rate=sample_rate)
```

Out[10]:
0:00 / 0:00

The entire word can be heard. It is impossible to cut all the files manually and do this basing on the simple plot. But you can use for example *webrtcvad* package which has a good VAD.

We can plot the spectrogram, together with presumed alignment of 'y' 'e' 's' graphemes

```

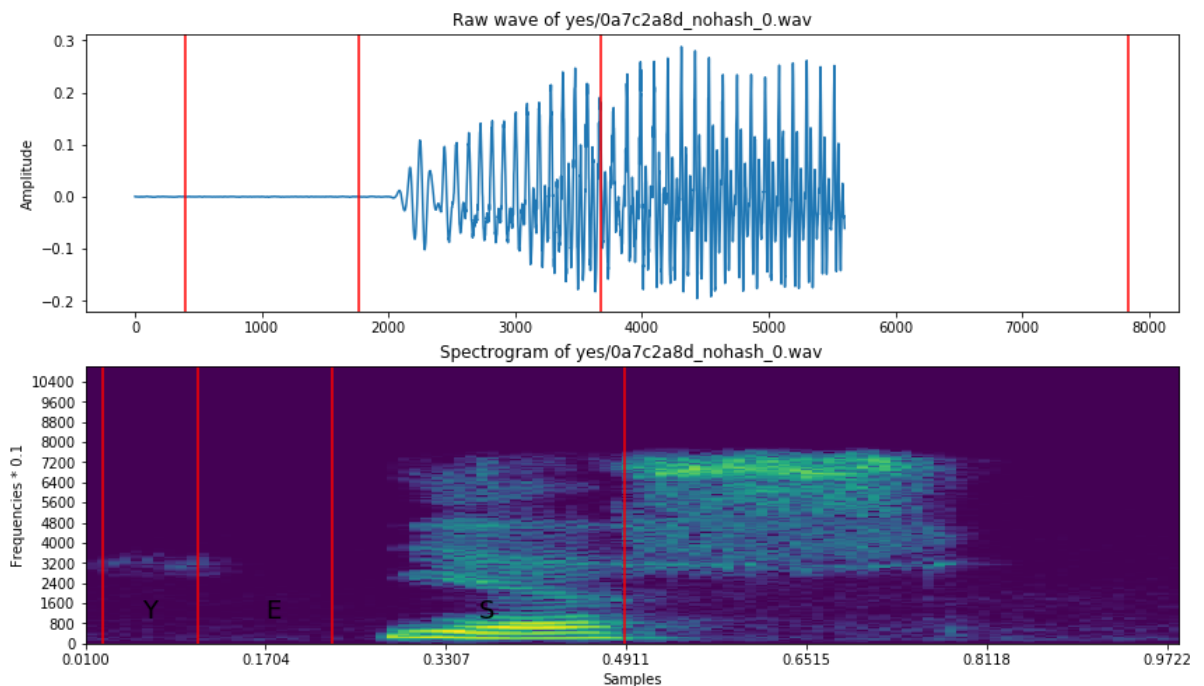
In [11]: freqs, times, spectrogram_cut = log_spectrogram(samples, sample_rate)

fig = plt.figure(figsize=(14, 8))
ax1 = fig.add_subplot(211)
ax1.set_title('Raw wave of ' + filename)
ax1.set_ylabel('Amplitude')
ax1.plot(samples_cut)

ax2 = fig.add_subplot(212)
ax2.set_title('Spectrogram of ' + filename)
ax2.set_ylabel('Frequencies * 0.1')
ax2.set_xlabel('Samples')
ax2.imshow(spectrogram_cut.T, aspect='auto', origin='lower',
           extent=[times.min(), times.max(), freqs.min(), freqs.max()])
ax2.set_yticks(freqs[::16])
ax2.set_xticks(times[::16])
ax2.text(0.06, 1000, 'Y', fontsize=18)
ax2.text(0.17, 1000, 'E', fontsize=18)
ax2.text(0.36, 1000, 'S', fontsize=18)

xcoords = [0.025, 0.11, 0.23, 0.49]
for xc in xcoords:
    ax1.axvline(x=xc*16000, c='r')
    ax2.axvline(x=xc, c='r')

```



Please adjust boundaries of sounds for 'Y', 'E' and 'S' in the above code.

Resampling - dimensionality reduction

Another way to reduce the dimensionality of our data is to resample recordings.

You can hear that the recording don't sound very natural, because they are sampled with 16k frequency, and we usually hear much more. However, [the most speech related frequencies are presented in smaller band](https://en.wikipedia.org/wiki/Voice_frequency) (https://en.wikipedia.org/wiki/Voice_frequency). That's why you can still understand another person talking to the telephone, where GSM signal is sampled to 8000 Hz.

Summarizing, we could resample our dataset to 8k. We will discard some information that shouldn't be important, and we'll reduce size of the data.

We have to remember that it can be risky, because sometimes very small difference in sound matter, so we don't want to lost anything. On the other hand, first experiments can be done much faster with smaller training size.

We'll need to calculate FFT (Fast Fourier Transform). Definition:

```
In [43]: def custom_fft(y, fs):  
    T = 1.0 / fs  
    N = y.shape[0]  
    yf = fft(y)  
    xf = np.linspace(0.0, 1.0/(2.0*T), N//2)  
    vals = 2.0/N * np.abs(yf[0:N//2]) # FFT is simmetrical, so we take just t  
he first half  
    # FFT is also complex, to we take just the real part (abs)  
    return xf, vals
```

Let's read some recording, resample it, and listen. We can also compare FFT-s. Notice, that there is almost no information above 4000 Hz in the original signal.

```
In [13]: audio_path='./data'  
filename = '/happy/0b09edd3_nohash_0.wav'  
new_sample_rate = 16000  
  
samples, sample_rate = librosa.load(str(audio_path) + filename)  
resampled = signal.resample(samples, int(new_sample_rate/sample_rate * samples  
.shape[0]))
```

```
In [14]: ipd.Audio(samples, rate=sample_rate)
```

Out[14]:

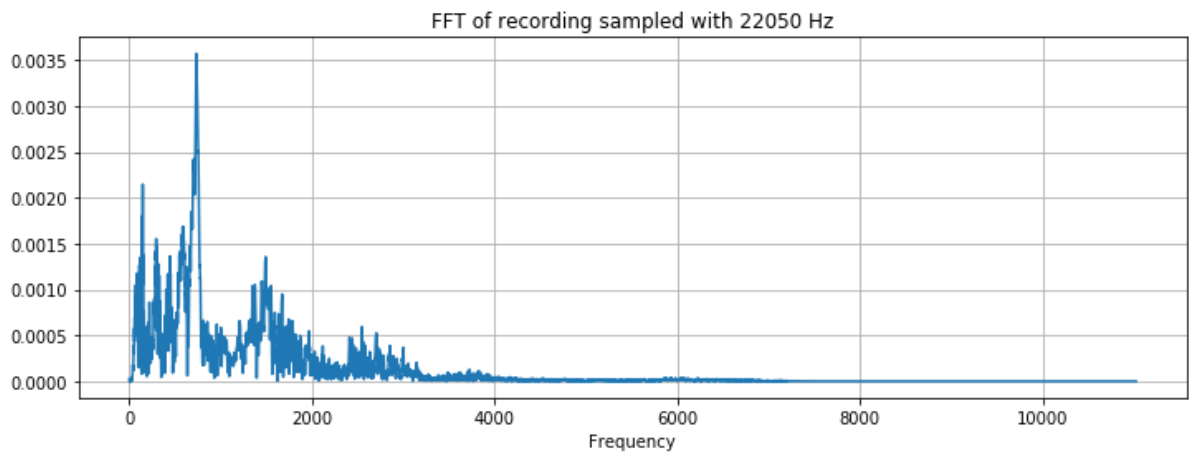
0:00 / 0:01

```
In [16]: ipd.Audio(resampled, rate=new_sample_rate)
```

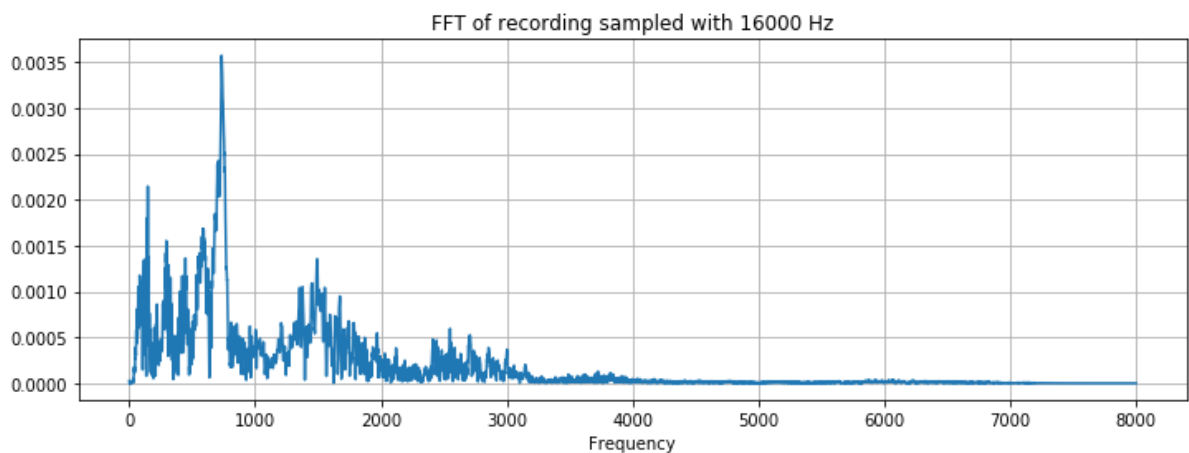
```
Out[16]:  
0:00 / 0:01
```

There is almost no difference!

```
In [15]: xf, vals = custom_fft(samples, sample_rate)  
plt.figure(figsize=(12, 4))  
plt.title('FFT of recording sampled with ' + str(sample_rate) + ' Hz')  
plt.plot(xf, vals)  
plt.xlabel('Frequency')  
plt.grid()  
plt.show()
```



```
In [17]: xf, vals = custom_fft(resampled, new_sample_rate)  
plt.figure(figsize=(12, 4))  
plt.title('FFT of recording sampled with ' + str(new_sample_rate) + ' Hz')  
plt.plot(xf, vals)  
plt.xlabel('Frequency')  
plt.grid()  
plt.show()
```



By resampling our wav files with twice lower frequency, we reduce the volume of Fourier image by 2.

Features extraction steps

We will use the following feature extraction algorithm :

1. Resampling to reduce the volume of data
2. *VAD (Voice Activated Detection)* for elimination or padding of portion of files without speech.
3. Maybe padding with 0 to make all signals have equal length
4. Determine Log spectrogram (or *MFCC*, or *PLP*)
5. Features normalization with *mean* and *std*
6. Stacking of a given number of frames to get temporal information

Perceptual Linear Prediction (PLP) Motivated by hearing perception, it uses equal loudness pre-emphasis and cube-root compression instead of the log compression. It also uses linear regression to finalize the cepstral coefficients. PLP has slightly better accuracy and slightly better noise robustness.

Dataset examination

Before we proceed with extraction of features with need to perform standard investigation of the dataset.

Identify the Number of Labels, Numbers of Samples under each label, etc.

```
In [38]: audio_path= './data'
         dirs = [f for f in os.listdir(audio_path) if isdir(join(audio_path,f))]
         dirs.sort()
         print('Number of labels: ' + str(len(dirs)))
```

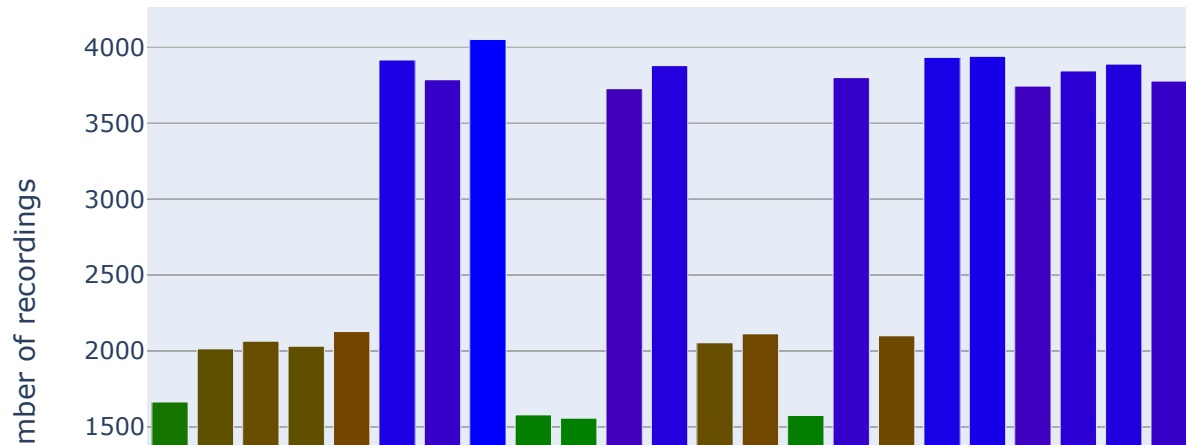
Number of labels: 35

Histogram with number of samples in every class (label)

```
In [39]: """
HW Assignment"""
number_of_recordings = []
for direct in dirs:
    waves = [f for f in os.listdir(join(audio_path, direct)) if f.endswith('.wav')]
    number_of_recordings.append(len(waves))

data = [go.Histogram(x=dirs, y=number_of_recordings)]
trace = go.Bar(
    x=dirs,
    y=number_of_recordings,
    marker=dict(color = number_of_recordings, colorscale=[[0, 'green'], [0.5,
'red'], [1.0, 'rgb(0, 0, 255)']], showscale=True
),
)
layout = go.Layout(
    title='Number of recordings in given label',
    xaxis = dict(title='Words'),
    yaxis = dict(title='Number of recordings')
)
py.iplot(go.Figure(data=[trace], layout=layout))
"""
```

Number of recordings in given label



Out[39]: ''

Dataset is not uniform but is balanced. There are no words (labels) with number of samples orders of magnitude smaller than in other classes.

Deeper Analysis of the Recordings

Recordings come from very different sources. Some of them can come from mobile GSM channel.

It is extremely important to split the dataset in a way that one speaker doesn't occur in both train and test sets.

We will actually not perform such separation here. We do not have enough meta data and spektral analysis for classiffication of sounds by speakers is longer than we can tolerate in this set of examples.

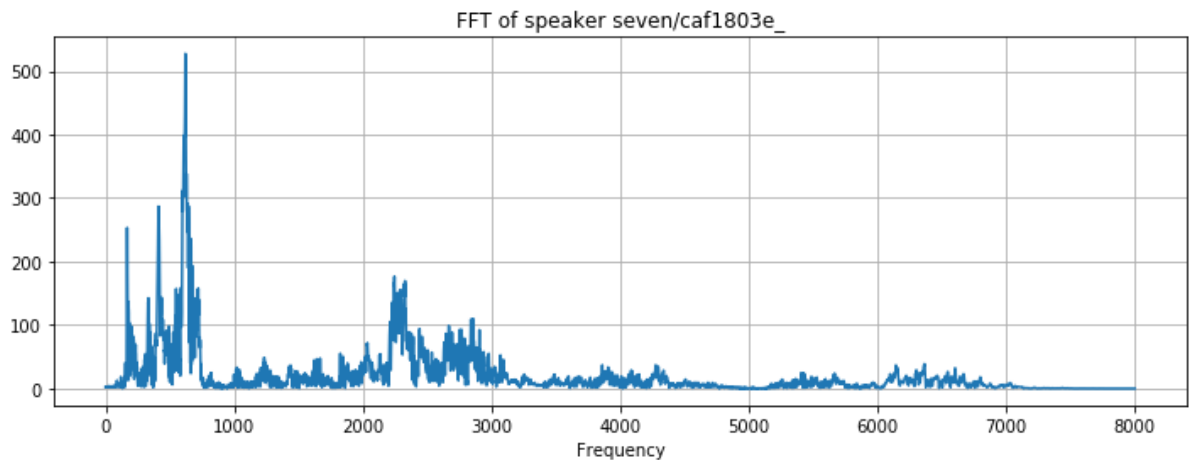
To see that there are repeated sampels coming from the same speaker, look and listen to this two examlpes:

```
In [21]: audio_path="data/"
# filenames = ['/on/004ae714_nohash_1.wav', '/on/0137b3f4_nohash_1.wav']
filenames = ['seven/caf1803e_nohash_0.wav', 'seven/0b77ee66_nohash_1.wav']
for filename in filenames:
    print(filename)
    print(audio_path)
    print(join(audio_path, filename))
    sample_rate, samples = wavfile.read(join(audio_path, filename))
    xf, vals = custom_fft(samples, sample_rate)
    plt.figure(figsize=(12, 4))
    plt.title('FFT of speaker ' + filename[0:15])
    plt.plot(xf, vals)
    plt.xlabel('Frequency')
    plt.grid()
    plt.show()
```

seven/caf1803e_nohash_0.wav

data/

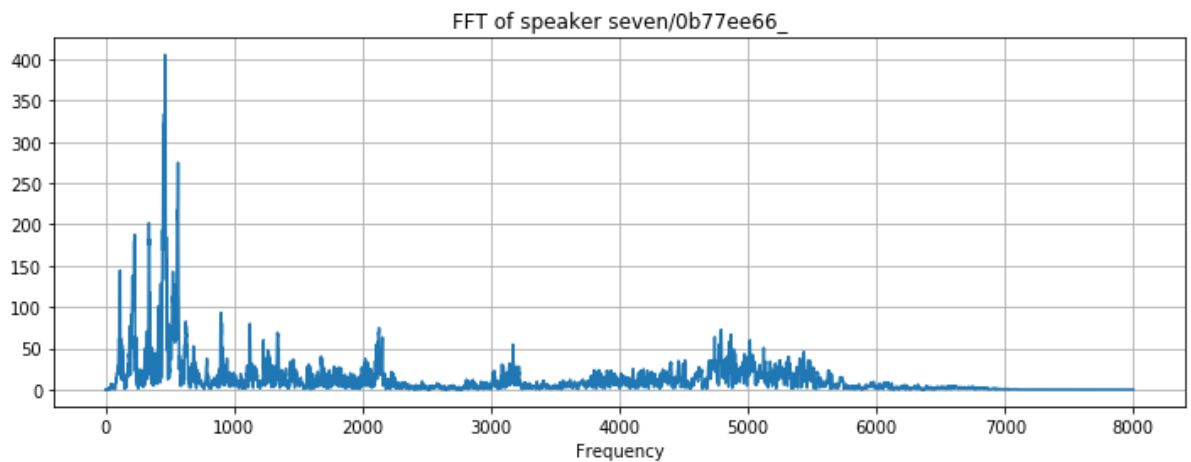
data/seven/caf1803e_nohash_0.wav



seven/0b77ee66_nohash_1.wav

data/

data/seven/0b77ee66_nohash_1.wav



It is even better to listen to the sound.

```
In [22]: print('Speaker ' + filenames[0][0:15])
         ipd.Audio(str(audio_path)+filenames[0])
         ipd.Audio(samples, rate=sample_rate)
```

Speaker seven/caf1803e_

Out[22]:
0:00 / 0:01

```
In [23]: ipd.Audio(samples, rate=sample_rate)
```

Out[23]:
0:00 / 0:01

The next one comes from a different speaker.

```
In [24]: print('Speaker ' + filenames[0][0:15])
         #ipd.Audio(join(train_audio_path, filenames[1]))
         print('seven/d21fd169_nohash_1.wav')
         ipd.Audio(join(audio_path, 'seven/d21fd169_nohash_1.wav'))
```

Speaker seven/caf1803e_
seven/d21fd169_nohash_1.wav

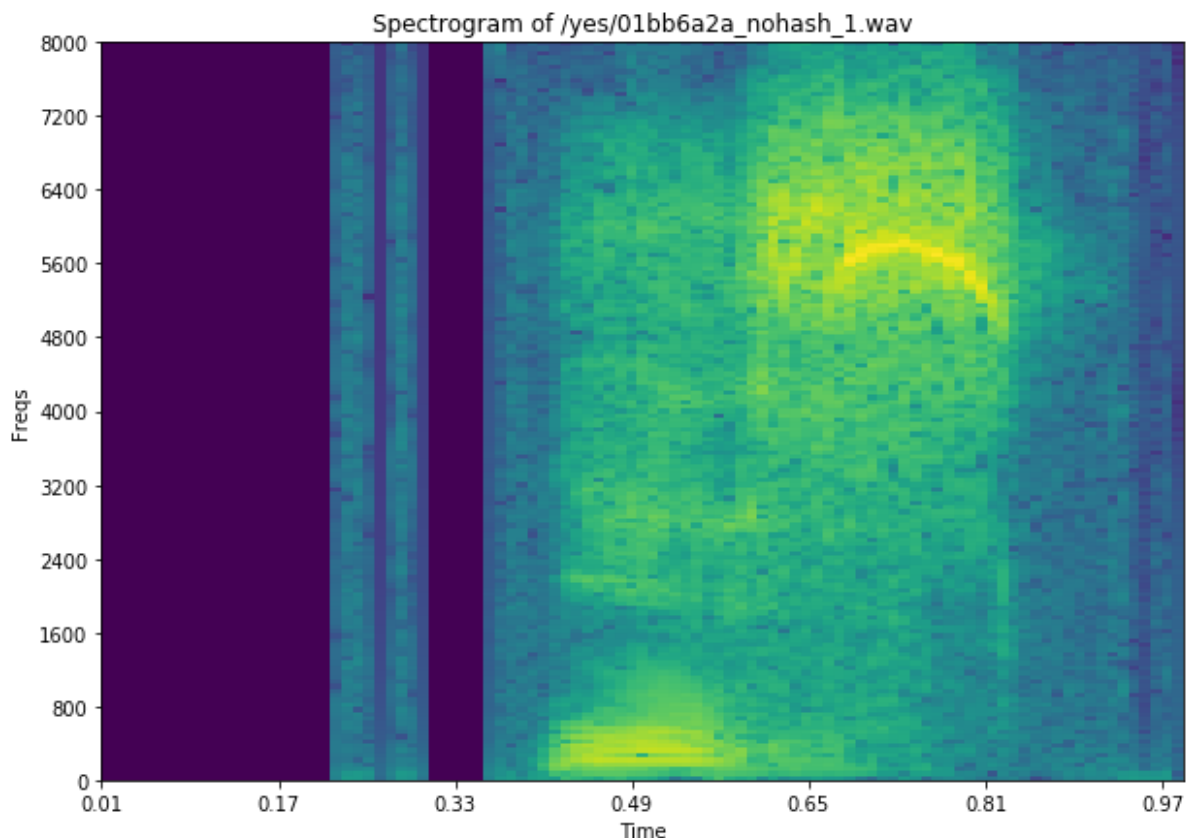
Out[24]:
0:00 / 0:01

Removal of Silence

There are also recordings with noticeable silence. Silence in captured recording is an opportunity for some compression?

```
In [25]: filename = '/yes/01bb6a2a_nohash_1.wav'
sample_rate, samples = wavfile.read(str(audio_path) + filename)
freqs, times, spectrogram = log_spectrogram(samples, sample_rate)

plt.figure(figsize=(10, 7))
plt.title('Spectrogram of ' + filename)
plt.ylabel('Freqs')
plt.xlabel('Time')
plt.imshow(spectrogram.T, aspect='auto', origin='lower',
           extent=[times.min(), times.max(), freqs.min(), freqs.max()])
plt.yticks(freqs[::16])
plt.xticks(times[::16])
plt.show()
```



Dark blue regions represent silence, i.e. absence of useful signal.

Determine lengths of recordings

As an illustration we will find all the files that have 1 second duration:

```
In [26]: print(dirs)
print(audio_path)

['backward', 'bed', 'bird', 'cat', 'dog', 'down', 'eight', 'five', 'follow',
'forward', 'four', 'go', 'happy', 'house', 'learn', 'left', 'marvin', 'nine',
'no', 'off', 'on', 'one', 'right', 'seven', 'sheila', 'six', 'stop', 'three',
'tree', 'two', 'up', 'visual', 'wow', 'yes', 'zero']
data/
```

```
In [27]: num_of_shorter = 0
print(audio_path)
for direct in dirs:
    waves = [f for f in os.listdir(str(audio_path)+ direct) if f.endswith('.wav')]
    for wav in waves:
        sample_rate, samples = wavfile.read(str(audio_path) +direct + '/' + wav)
        if samples.shape[0] < sample_rate:
            num_of_shorter += 1
print('Number of recordings shorter than 1 second: ' + str(num_of_shorter))

data/
Number of recordings shorter than 1 second: 10435
```

To bring those short files up to the standard length, we can pad them with zeros. Our Deep Learning techniques usually prefer if we feed them inputs of uniform size.

Mean Spectrograms and FFT

Let us plot mean FFT for several words. We will select words: 'yes', 'bed', 'cat', 'happy' & 'seven'

```

In [29]: to_keep = 'yes bed cat happy seven'.split()
        dirs = [d for d in dirs if d in to_keep]

        print(dirs)

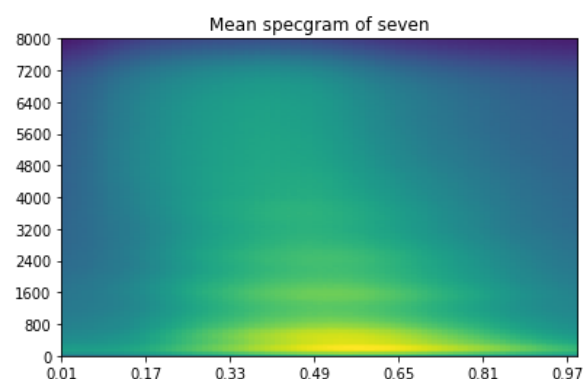
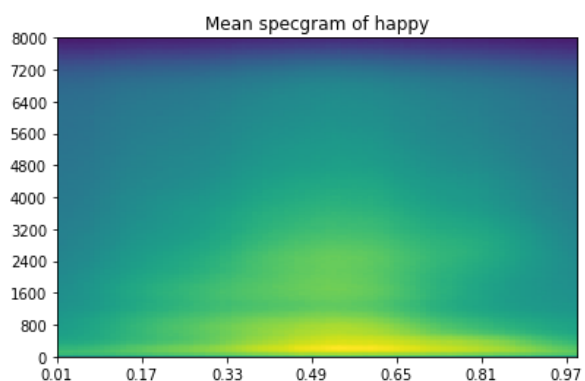
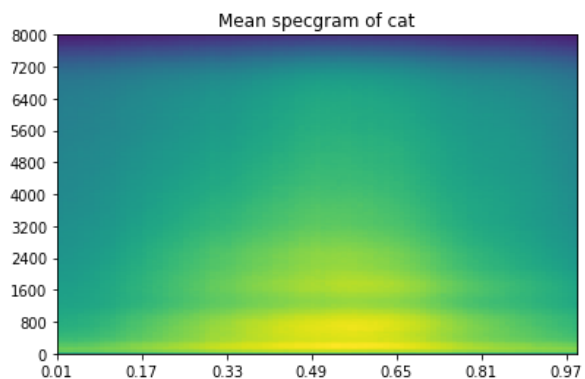
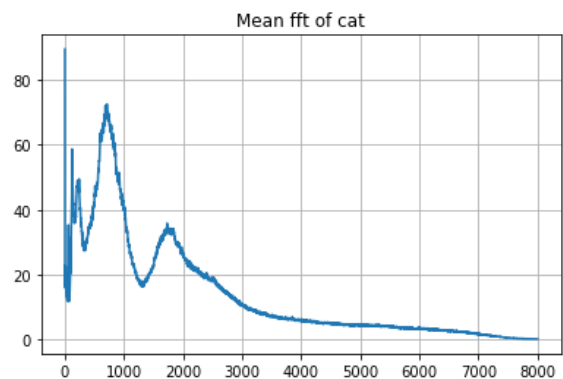
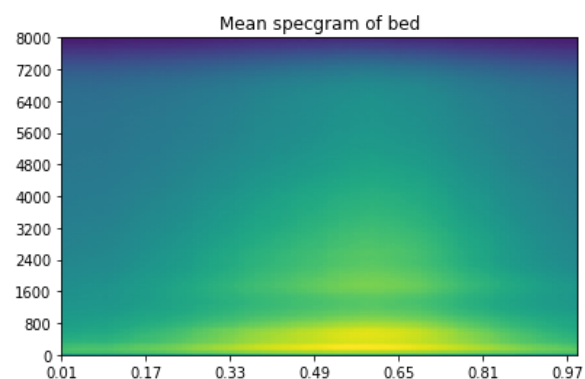
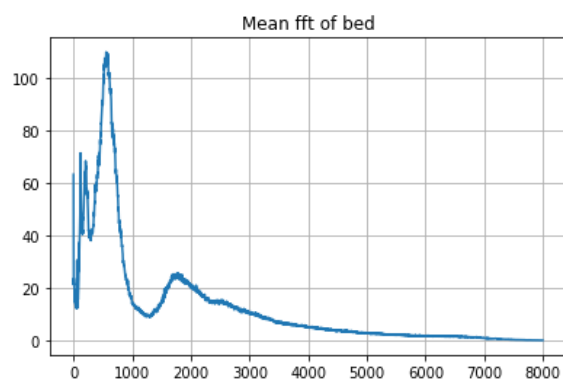
        for direct in dirs:
            vals_all = []
            spec_all = []

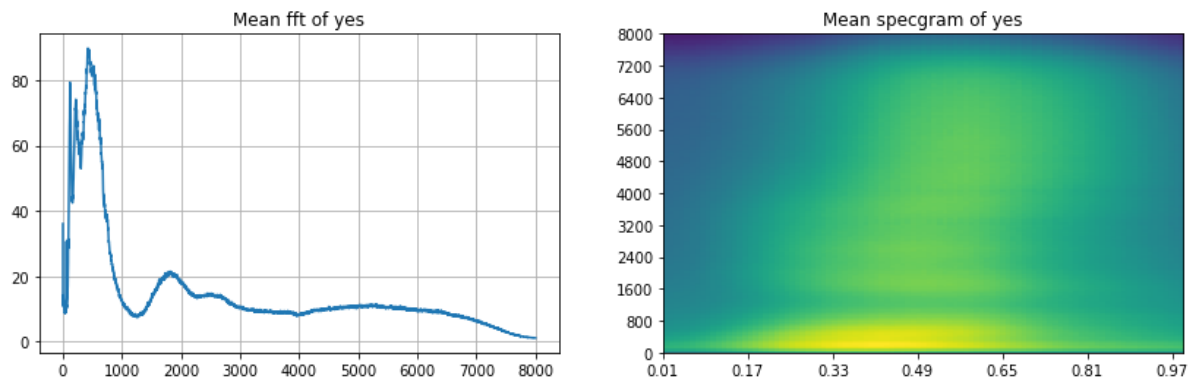
            waves = [f for f in os.listdir(join(audio_path, direct)) if f.endswith('.wav')]
            for wav in waves:
                sample_rate, samples = wavfile.read(audio_path + '/' + direct + '/' + wav)
                if samples.shape[0] != 16000:
                    continue
                xf, vals = custom_fft(samples, 16000)
                vals_all.append(vals)
                freqs, times, spec = log_specgram(samples, 16000)
                spec_all.append(spec)

            plt.figure(figsize=(14, 4))
            plt.subplot(121)
            plt.title('Mean fft of ' + direct)
            plt.plot(np.mean(np.array(vals_all), axis=0))
            plt.grid()
            plt.subplot(122)
            plt.title('Mean specgram of ' + direct)
            plt.imshow(np.mean(np.array(spec_all), axis=0).T, aspect='auto', origin='lower',
                       extent=[times.min(), times.max(), freqs.min(), freqs.max()])
            plt.yticks(freqs[::16])
            plt.xticks(times[::16])
            plt.show()

```


['bed', 'cat', 'happy', 'seven', 'yes']





Gaussian Mixtures modeling

We can see that mean FFT looks different for every word. We could model each FFT with a mixture of Gaussian distributions. Some of them however, look almost identical on FFT, like *stop* and *up*... They are still distinguishable when we look at spectrograms! High frequencies are earlier than low at the beginning of *stop* (probably s).

That's why temporal component is also necessary. There is a [Kaldi](http://kaldi-asr.org/) (<http://kaldi-asr.org/>) library, that can model words (or smaller parts of words) with GMMs and model temporal dependencies with [Hidden Markov Models](https://github.com/danijel3/ASRDemos/blob/master/notebooks/HMM_FST.ipynb) (https://github.com/danijel3/ASRDemos/blob/master/notebooks/HMM_FST.ipynb).

We could use simple GMMs for words to check what can we model and how hard it is to distinguish the words. We can use [Scikit-learn](http://scikit-learn.org/) (<http://scikit-learn.org/>) for that, however it is not straightforward and lasts very long here, so I abandon this idea for now.

Frequency components across the words

One popular type of plots that depicts the frequency content of different words is called violin plot.

```

In [30]: def violinplot_frequency(dirs, freq_ind):
        """ Plot violinplots for given words (waves in dirs) and frequency freq_in
        d
        from all frequencies freqs. """

        spec_all = [] # Contain spectrograms
        ind = 0
        for direct in dirs:
            spec_all.append([])

            waves = [f for f in os.listdir(join(audio_path, direct)) if
                      f.endswith('.wav')]
            for wav in waves[:100]:
                sample_rate, samples = wavfile.read(
                    audio_path + '/' + direct + '/' + wav)
                freqs, times, spec = log_specgram(samples, sample_rate)
                spec_all[ind].extend(spec[:, freq_ind])
            ind += 1

        # Different lengths = different num of frames. Make number equal
        minimum = min([len(spec) for spec in spec_all])
        spec_all = np.array([spec[:minimum] for spec in spec_all])

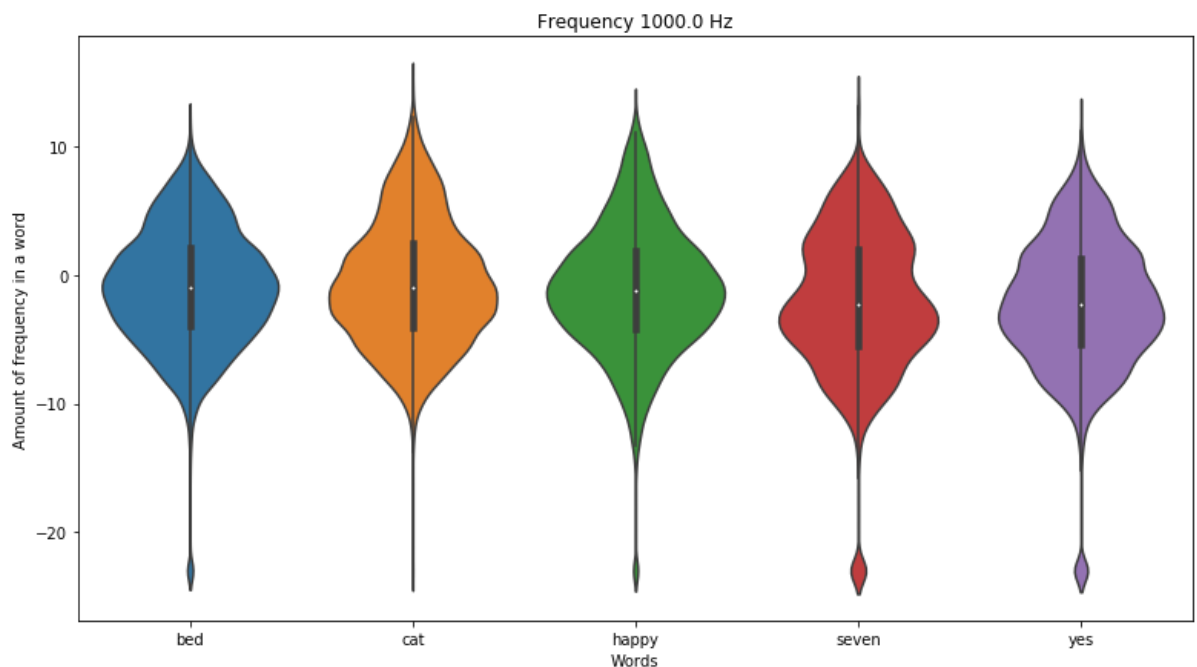
        plt.figure(figsize=(13,7))
        plt.title('Frequency ' + str(freqs[freq_ind]) + ' Hz')
        plt.ylabel('Amount of frequency in a word')
        plt.xlabel('Words')
        sns.violinplot(data=pd.DataFrame(spec_all.T, columns=dirs))
        plt.show()

```

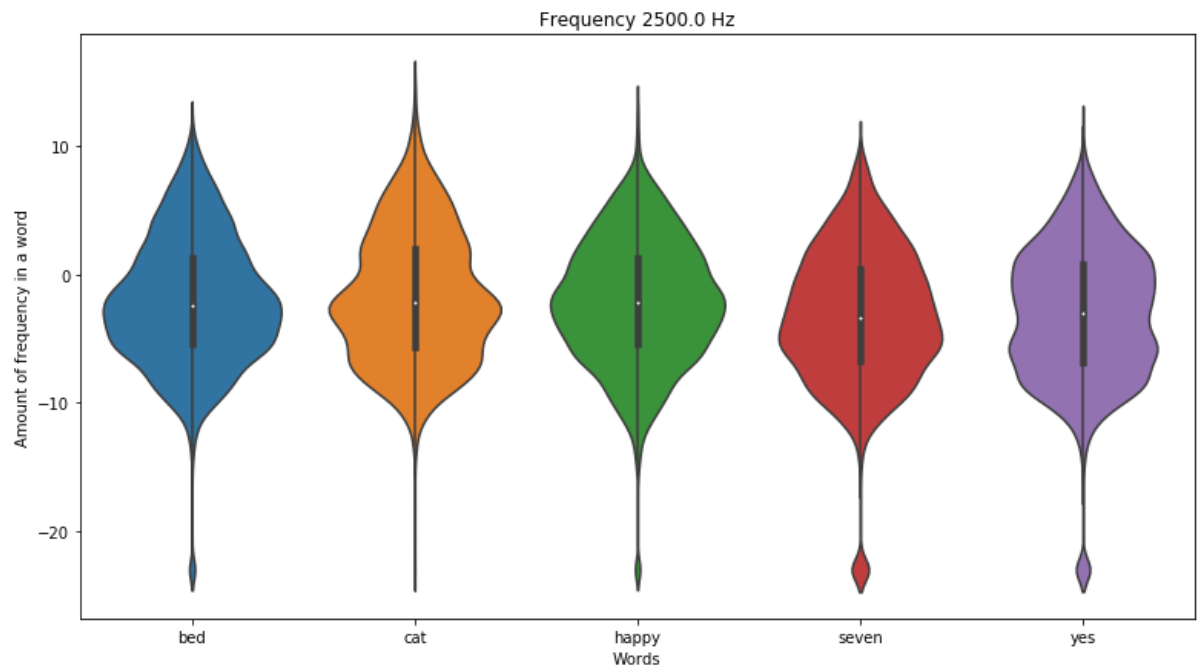
```

In [31]: violinplot_frequency(dirs, 20)

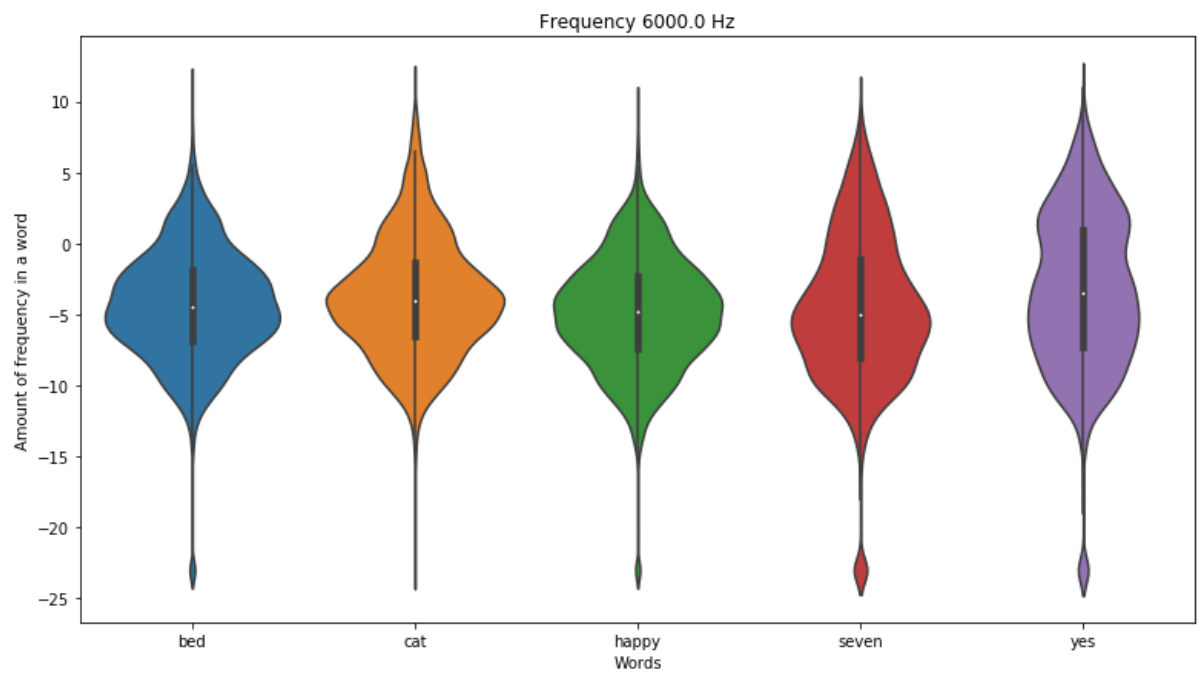
```



```
In [32]: violinplot_frequency(dirs, 50)
```



```
In [33]: violinplot_frequency(dirs, 120)
```



Anomaly detection

We should check if there are any recordings that somehow stand out from the rest. We can project our entire dataset to a space of lower dimensionality (2 or 3) and interactively check for any anomaly. We will use PCA for dimensionality reduction.

Please note that the following cell takes long time to run. To reduce run time, we reduced the number of words in ./data directory to six: bed , bird , cat , five , happy and seven

```
In [40]: audio_path= './data'
        dirs = [f for f in os.listdir(audio_path) if isdir(join(audio_path,f))]
        dirs.sort()
        print('Number of labels: ' + str(len(dirs)))
```

Number of labels: 6

```

In [44]: fft_all = []
names = []
for direct in dirs:
    waves = [f for f in os.listdir(str(audio_path) + '/' + direct) if f.endswith(
        '.wav')]
    for wav in waves:
        samples, sample_rate = librosa.load(audio_path + '/' + direct + '/' + wav)
        if samples.shape[0] != sample_rate:
            samples = np.append(samples, np.zeros((sample_rate - samples.shape
            [0], )))
        x, val = custom_fft(samples, sample_rate)
        fft_all.append(val)
        names.append(direct + '/' + wav)

fft_all = np.array(fft_all)

# Normalization
fft_all = (fft_all - np.mean(fft_all, axis=0)) / np.std(fft_all, axis=0)

# Dim reduction
pca = PCA(n_components=3)
fft_all = pca.fit_transform(fft_all)

def interactive_3d_plot(data, names):
    scatt = go.Scatter3d(x=data[:, 0], y=data[:, 1], z=data[:, 2], mode='markers', text=names)
    data = go.Data([scatt])
    layout = go.Layout(title="Anomaly detection")
    figure = go.Figure(data=data, layout=layout)
    py.iplot(figure)

interactive_3d_plot(fft_all, names)

```

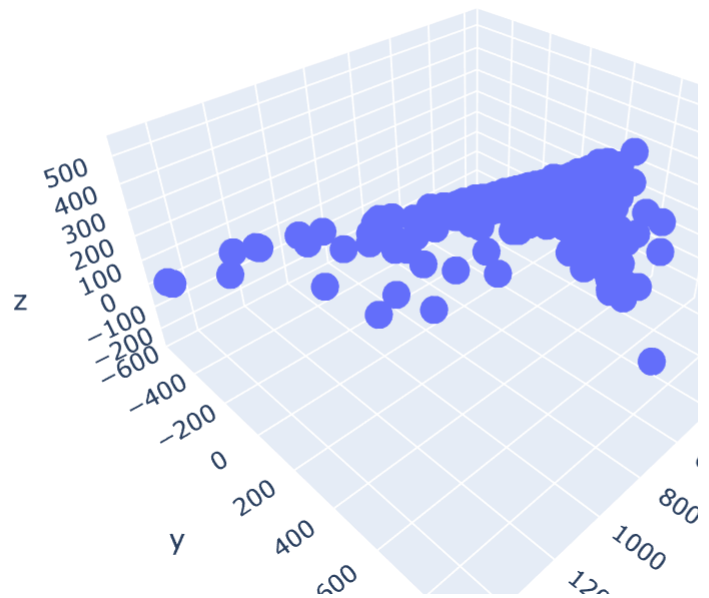
C:\ProgramData\Anaconda3\lib\site-packages\plotly\graph_objs_deprecations.py:40: DeprecationWarning:

plotly.graph_objs.Data is deprecated.

Please replace it with a list or tuple of instances of the following types

- plotly.graph_objs.Scatter
- plotly.graph_objs.Bar
- plotly.graph_objs.Area
- plotly.graph_objs.Histogram
- etc.

Anomaly detection



Please notice that you can rotate the above 3-d plot and expose outliers in different portions of this projection space. Notice that there are `yes/e4b02540_nohash_0.wav`, `happy/abbfc3b4_nohash_1.wav`, `bed/9aa21fa9_nohash_0.wav` and more points lie far away from the rest. Those are outliers or anomalies. Listen to those outliers. You will hear that some of them sound unusual. For others it is hard to tell what is "wrong" with them.

```
In [ ]: print('Recording of happy/abbfc3b4_nohash_1.wav')
        ipd.Audio(audio_path + '/' + 'happy/abbfc3b4_nohash_1.wav')
```

```
In [ ]: print('Recording yes/e4b02540_nohash_0.wav')
        ipd.Audio(join(audio_path + '/' + 'yes/e4b02540_nohash_0.wav'))
```

If you are looking for anomalies for individual words, you could, for example, find this file for word seven:

```
In [37]: print('Recording seven/b1114e4f_nohash_0.wav')
         ipd.Audio(audio_path + '/' + 'seven/b1114e4f_nohash_0.wav')
```

Recording seven/b1114e4f_nohash_0.wav

Out[37]:

0:00 / 0:01

The first two samples contained nothing obviously important. The third sample did sound weird. Usually, you can find some distortions using this method.

Where to look for the inspiration

Speech recognition is a really big R&D field and it is hard to get to know important things in short time!

You can take many different approaches for solutions based on neural networks and architectures. Connectionist Temporal Classification (CTC) by using Recurrent Neural Network (RNN) in TensorFlow, Keras, Automatic Speech Recognition (ASR), Hidden Markov Model (HMM) based on ASR, Gaussian mixture models with HMMs, Deep models with HMMs, Large Vocabulary Continuous Speech Recognition Systems (LVCSR).

This tutorial focuses on two models and a very small vocabulary, with recordings with only one word in it, with a (mostly) given length:

1. Encoder-decoder: <https://arxiv.org/abs/1508.01211> (<https://arxiv.org/abs/1508.01211>)
2. RNNs with CTC loss: <https://arxiv.org/abs/1412.5567> (<https://arxiv.org/abs/1412.5567>)
3. Classic speech recognition is described here:
<http://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/tutorial%20on%20hmm%20and%20applications.pdf>
<http://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/tutorial%20on%20hmm%20and%20applications.pdf>

You can find *Kaldi Tutorial for dummies* (http://kaldi-asr.org/doc/kaldi_for_dummies.html), with a problem similar to this competition in some way.

Notebook code was adapted from the following sources:

Tensorflow:

https://www.tensorflow.org/tutorials/audio_recognition (https://www.tensorflow.org/tutorials/audio_recognition)
<https://github.com/ugnelis/tensorflow-rnn-ctc/blob/master/README.md> (<https://github.com/ugnelis/tensorflow-rnn-ctc/blob/master/README.md>)

LibROSA: <https://github.com/librosa/librosa/blob/master/examples/LibROSA%20demo.ipynb>
<https://github.com/librosa/librosa/blob/master/examples/LibROSA%20demo.ipynb>

Kaggle / Data / Python code:

<https://github.com/pannous/tensorflow-speech-recognition> (<https://github.com/pannous/tensorflow-speech-recognition>) <https://www.kaggle.com/davids1992/speech-representation-and-data-exploration/notebook>
<https://www.kaggle.com/davids1992/speech-representation-and-data-exploration/notebook>
<https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/download/train.7z>
<https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/download/train.7z>
<https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/download/test.7z>
<https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/download/test.7z>



Sampled Sound Data Processing

We now know how to generate an array of numbers which represent the sound wave's amplitude at 1/16.000 of a second intervals.

We could feed these numbers right into a neural network. But trying to recognize speech patterns by processing these samples directly is difficult. Instead, we can make the problem easier by doing some pre-processing on the audio data. Let's start by grouping our sampled audio into 20-millisecond-long chunks. Here's our first 20 milliseconds of audio (i.e., our first 400 samples): Plotting those numbers as a simple line graph gives us a rough approximation of the original sound wave for that 20 millisecond period of time. This recording is only 1/50th of a second long. But even this short recording is a complex mish-mash of different frequencies of sound. There's some low sounds, some mid-range sounds, and even some high-pitched sounds sprinkled in. But taken all together, these different frequencies mix together to make up the complex sound of human speech.

To make this data easier for a neural network to process, we are going to break apart this complex sound wave into it's component parts. We'll break out the low-pitched parts, the next-lowest-pitched-parts, and so on. Then by adding up how much energy is in each of those frequency bands (from low to high), we create a fingerprint of sorts for this audio snippet.

We do this using a mathematic operation called a Fourier transform. It breaks apart the complex sound wave into the simple sound waves that make it up. Once we have those individual sound waves, we add up how much energy is contained in each one.

The end result is a score of how important each frequency range is, from low pitch (i.e. bass notes) to high pitch. Each number below represents how much energy was in each 50hz band of our 20 millisecond audio clip. But this is a lot easier to see when you draw this as a chart. If we repeat this process on every 20 millisecond chunk of audio, we end up with a spectrogram (each column from left-to-right is one 20ms chunk).

Simple Speech Recognition Demo

Code used in the example below is borrowed from this blog: <https://blog.manash.me/building-a-dead-simple-word-recognition-engine-using-convnet-in-keras-25e72c19c12b> (<https://blog.manash.me/building-a-dead-simple-word-recognition-engine-using-convnet-in-keras-25e72c19c12b>) . Processing in this portion of the notebook depends on the Python script preprocess.py. File preprocess.py is provide on the class site, week 12 folder.

Load Sound Datasets - *.wav files

First step will be to read the audio file from given path. The task will be to classify audio files for three classes: bed, cat and happy

There are basically three classes to recognize sounds. You can add as much as you may wish by downloading data from Kaggle if you are willing to classify more than these three. Just change the number of output at the softmax layer and you'll be good to go.

Each folder contains approximately 1700 audio files. The name of the folder is actually the label of those audio files. You can play some audio files randomly to get an overall idea. This demo takes audio input from one channel only.

Build Model - Keras Embedding, Vectors

We need to prepare a fixed size vector for each audio file for classification. TensorFlow guide contains a very good guide on embedding. Embedding is a mapping from discrete objects, such as words, to vectors of real numbers. https://www.tensorflow.org/programmers_guide/embedding (https://www.tensorflow.org/programmers_guide/embedding). We are dealing with a specific domain, audio embedding. What technique we should actually use to embed the audio into vector space? **MFCC encoding** We then compute MFCC using **librosa** library.

MFCC vectors might vary in size for different audio input, remember CNNs can't handle sequence data so we need to prepare a fixed size vector for all of the audio files. To overcome this problem all we need to do is to pad the output vectors with constant value such as "zero"

For each audio file the vector is a 20 by 11 dimensional matrix. When we embed things, everything gets converted into vector (We can treat it just like an image). we need to reshape it to give a depth. we need to encode this output vector into one-hot-encoded one to perform softmax (keras has a built in function)

Save data to array file first 11 x 20 dimensions 3 classes: bed, cat, happy 50 epochs 100 batch size

```
In [ ]: !pip install ffmpeg
```

Package FFMPEG

FFmpeg is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play pretty much any format that humans and machines have created. FFmpeg supports the most obscure ancient formats and the most modern, cutting edge, multimedia format. Librosa apparently depends on FFmpeg.

Both on the command prompt and in a Jupyter cell install ffmpeg using pip, e.g.

```
!pip install ffmpeg
```

The above might not be enough.

From <https://ffmpeg.zeranoe.com/builds/> download `ffmpeg-20190426-f857753-win64-static.zip` or similarly named file and expand it in one of program directories, for example `C:\Program File\ffmpeg`. Add directory `C:\Program Files\ffmpeg\bin` (or whatever `ffmpeg/bin` directory you have) to your enviromental variable `PATH`. On Mac OS, your `PATH` woud read differently. On Ubuntu your directory is most probaly `/usr/bin/ffmpeg`. Verify by typing `which ffmpeg` at the Linux prompt.

Version of librosa package

Code below requires older version of librosa package. Please downgrade librosa to the version 0.6.3

```
In [ ]: !pip install --upgrade librosa==0.6.3
```

You need to run this cell once, transform Wav files into MFCC- and export transformed values into *.npy files*.
Training of the model with different words does nto require new set of .npy files.

```
In [2]: %load_ext autoreload
        #%autoreload 2
        import tensorflow as tf
        from preprocess import *
        from tensorflow import keras
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
        from tensorflow.keras.utils import to_categorical
        from scipy.io import wavfile
        import IPython.display as ipd

        # Second dimension of the feature is dim2
        feature_dim_2 = 11
        path = './data'
        # Save data to array file first
        save_data_to_array(max_len=feature_dim_2)
```

```
Saving vectors of label - 'backward': 100%|██████████  
██████████ | 1664/1664 [03:40<00:00, 7.55it/s]  
Saving vectors of label - 'bed': 100%|██████████  
██████████ | 2014/2014 [04:11<00:00, 8.02it/s]  
Saving vectors of label - 'bird': 100%|██████████  
██████████ | 2064/2064 [04:21<00:00, 7.89it/s]  
Saving vectors of label - 'cat': 100%|██████████  
██████████ | 2031/2031 [04:16<00:00, 7.93it/s]  
Saving vectors of label - 'dog': 100%|██████████  
██████████ | 2128/2128 [04:24<00:00, 8.03it/s]  
Saving vectors of label - 'down': 100%|██████████  
██████████ | 3917/3917 [08:09<00:00, 8.01it/s]  
Saving vectors of label - 'eight': 100%|██████████  
██████████ | 3787/3787 [07:50<00:00, 8.05it/s]  
Saving vectors of label - 'five': 100%|██████████  
██████████ | 4052/4052 [08:23<00:00, 8.05it/s]  
Saving vectors of label - 'follow': 100%|██████████  
██████████ | 1579/1579 [03:15<00:00, 8.08it/s]  
Saving vectors of label - 'forward': 100%|██████████  
██████████ | 1557/1557 [03:13<00:00, 8.05it/s]  
Saving vectors of label - 'four': 100%|██████████  
██████████ | 3728/3728 [07:42<00:00, 8.06it/s]  
Saving vectors of label - 'go': 100%|██████████  
██████████ | 3880/3880 [08:02<00:00, 8.05it/s]  
Saving vectors of label - 'happy': 100%|██████████  
██████████ | 2054/2054 [04:14<00:00, 8.06it/s]  
Saving vectors of label - 'house': 100%|██████████  
██████████ | 2113/2113 [04:21<00:00, 8.07it/s]  
Saving vectors of label - 'learn': 100%|██████████  
██████████ | 1575/1575 [03:15<00:00, 8.05it/s]  
Saving vectors of label - 'left': 100%|██████████  
██████████ | 3801/3801 [07:52<00:00, 8.05it/s]  
Saving vectors of label - 'marvin': 100%|██████████  
██████████ | 2100/2100 [04:22<00:00, 8.01it/s]  
Saving vectors of label - 'nine': 100%|██████████  
██████████ | 3934/3934 [08:10<00:00, 8.02it/s]  
Saving vectors of label - 'no': 100%|██████████  
██████████ | 3941/3941 [08:12<00:00, 8.01it/s]  
Saving vectors of label - 'off': 100%|██████████  
██████████ | 3745/3745 [07:47<00:00, 8.01it/s]  
Saving vectors of label - 'on': 100%|██████████  
██████████ | 3845/3845 [07:59<00:00, 8.02it/s]  
Saving vectors of label - 'one': 100%|██████████  
██████████ | 3890/3890 [08:06<00:00, 7.99it/s]  
Saving vectors of label - 'right': 100%|██████████  
██████████ | 3778/3778 [07:52<00:00, 8.00it/s]  
Saving vectors of label - 'seven': 100%|██████████  
██████████ | 3998/3998 [08:39<00:00, 7.70it/s]  
Saving vectors of label - 'sheila': 100%|██████████  
██████████ | 2022/2022 [04:11<00:00, 8.04it/s]  
Saving vectors of label - 'six': 100%|██████████  
██████████ | 3860/3860 [08:02<00:00, 8.00it/s]  
Saving vectors of label - 'stop': 100%|██████████  
██████████ | 3872/3872 [08:02<00:00, 8.03it/s]  
Saving vectors of label - 'three': 100%|██████████  
██████████ | 3727/3727 [07:44<00:00, 8.02it/s]  
Saving vectors of label - 'tree': 100%|██████████
```

```

[03:39<00:00, 8.02it/s]
Saving vectors of label - 'two': 100%|
[08:05<00:00, 8.00it/s]
Saving vectors of label - 'up': 100%|
[07:45<00:00, 8.00it/s]
Saving vectors of label - 'visual': 100%|
[03:18<00:00, 8.03it/s]
Saving vectors of label - 'wow': 100%|
[04:25<00:00, 7.98it/s]
Saving vectors of label - 'yes': 100%|
[08:27<00:00, 7.97it/s]
Saving vectors of label - 'zero': 100%|
[08:28<00:00, 7.96it/s]

```

Loading train set and test set

The routine below looks into the `./data` directory and select for loading those `*.npy` files which have corresponding directory (word) in `./data` directory. `./data` directory could contain empty subdirectories named after selected words.

```
In [11]: X_train, X_test, y_train, y_test = get_train_test()

# # Feature dimension
feature_dim_1 = 20
channel = 1
epochs = 1000
batch_size = 100
verbose = 1
num_classes = 6    # number of words used for training.

# Reshaping to perform 2D convolution
X_train = X_train.reshape(X_train.shape[0], feature_dim_1, feature_dim_2, channel)
X_test = X_test.reshape(X_test.shape[0], feature_dim_1, feature_dim_2, channel)

y_train_hot = to_categorical(y_train)
y_test_hot = to_categorical(y_test)
```

```
In [12]: print(X_train.shape, X_test.shape)
          print(y_train.shape, y_test.shape)

(9727, 20, 11, 1) (6486, 20, 11, 1)
(9727,) (6486,)
```

Each word is represented by 20 samples (feature_dim_1 = 20) along the time axis and 11 MFCC (the first 11) at each time point.

Each word is represented as an image (matrix, array) of dimensions 50x11.

When we use all 20 words in the data set the dimensions of above tensors are: (38832, 20, 11, 1) (25889, 20, 11, 1) (38832,) (25889,)

Define the Model

CNN & Rectified Linear Units, Conv2D reduced into a single vector of class scores(ConvNet have neurons arranged in 3 dimensions: width, height, depth)

Epoch 50/50 Adadelta. keras.optimizers 3112/3112 3s 985us/step - loss: 0.0133 - acc: 0.9958 - val_loss: 0.2580 - val_acc: 0.9480

We will use rectified linear units (**ReLU**s) for the hidden layers. A rectified linear unit has output 0 if the input is less than 0, and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input.

ReLU's machinery is more like a real neuron in your body.

ReLU activations are the simplest non-linear activation function you can use. When you get the input is positive, the derivative is just 1, so there isn't the squeezing effect you meet on backpropagated errors from the sigmoid function. Research has shown that ReLUs result in much faster training for large networks.

```
In [5]: def get_model():
        model = Sequential()
        model.add(Conv2D(32, kernel_size=(2, 2), activation='relu', input_shape=(feature_dim_1, feature_dim_2, channel)))
        model.add(Conv2D(48, kernel_size=(2, 2), activation='relu'))
        model.add(Conv2D(120, kernel_size=(2, 2), activation='relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.25))
        model.add(Flatten())
        model.add(Dense(128, activation='relu'))
        model.add(Dropout(0.25))
        model.add(Dense(64, activation='relu'))
        model.add(Dropout(0.4))
        model.add(Dense(num_classes, activation='softmax'))
        model.compile(loss=keras.losses.categorical_crossentropy,
                      optimizer=keras.optimizers.Adadelta(),
                      metrics=['accuracy'])
        return model

# Predicts one sample
def predict(filepath, model):
    sample = wav2mfcc(filepath)
    sample_resaped = sample.reshape(1, feature_dim_1, feature_dim_2, channel)
    return get_labels()[0][
        np.argmax(model.predict(sample_resaped))
    ]
```


Train the Model

```
In [13]: model = get_model()
# model.fit(X_train, y_train_hot, batch_size=batch_size, epochs=epochs, verbose=verbose, validation_data=(X_test, y_test_hot))
hist = model.fit(X_train, y_train_hot, batch_size=batch_size, epochs=epochs, verbose=verbose, validation_data=(X_test, y_test_hot))
```

Train on 9727 samples, validate on 6486 samples

Epoch 1/1000

9727/9727 [=====] - 1s 97us/sample - loss: 5.8685 - accuracy: 0.1588 - val_loss: 2.8450 - val_accuracy: 0.1617

Epoch 2/1000

9727/9727 [=====] - 1s 54us/sample - loss: 5.0847 - accuracy: 0.1655 - val_loss: 2.4581 - val_accuracy: 0.1856

Epoch 3/1000

9727/9727 [=====] - 1s 54us/sample - loss: 4.6653 - accuracy: 0.1739 - val_loss: 2.2431 - val_accuracy: 0.1997

Epoch 4/1000

9727/9727 [=====] - 1s 54us/sample - loss: 4.3640 - accuracy: 0.1726 - val_loss: 2.1183 - val_accuracy: 0.2128

Epoch 5/1000

9727/9727 [=====] - 1s 54us/sample - loss: 4.0193 - accuracy: 0.1812 - val_loss: 2.0319 - val_accuracy: 0.2253

Epoch 6/1000

9727/9727 [=====] - 1s 54us/sample - loss: 3.7145 - accuracy: 0.1868 - val_loss: 1.9653 - val_accuracy: 0.2333

Epoch 7/1000

9727/9727 [=====] - 1s 54us/sample - loss: 3.5513 - accuracy: 0.1791 - val_loss: 1.9097 - val_accuracy: 0.2451

Epoch 8/1000

9727/9727 [=====] - 1s 54us/sample - loss: 3.3065 - accuracy: 0.2005 - val_loss: 1.8682 - val_accuracy: 0.2545

Epoch 9/1000

9727/9727 [=====] - 1s 54us/sample - loss: 3.1679 - accuracy: 0.1915 - val_loss: 1.8337 - val_accuracy: 0.2655

Epoch 10/1000

9727/9727 [=====] - 1s 56us/sample - loss: 3.0462 - accuracy: 0.1967 - val_loss: 1.8034 - val_accuracy: 0.2771

Epoch 11/1000

9727/9727 [=====] - 1s 54us/sample - loss: 2.8750 - accuracy: 0.2002 - val_loss: 1.7785 - val_accuracy: 0.2846

Epoch 12/1000

9727/9727 [=====] - 1s 54us/sample - loss: 2.8082 - accuracy: 0.1939 - val_loss: 1.7601 - val_accuracy: 0.2916

Epoch 13/1000

9727/9727 [=====] - 1s 54us/sample - loss: 2.6575 - accuracy: 0.2002 - val_loss: 1.7453 - val_accuracy: 0.2977

Epoch 14/1000

9727/9727 [=====] - 1s 54us/sample - loss: 2.5957 - accuracy: 0.2002 - val_loss: 1.7352 - val_accuracy: 0.3054

Epoch 15/1000

9727/9727 [=====] - 1s 54us/sample - loss: 2.5024 - accuracy: 0.1994 - val_loss: 1.7244 - val_accuracy: 0.3077

Epoch 16/1000

9727/9727 [=====] - 1s 54us/sample - loss: 2.4170 - accuracy: 0.2064 - val_loss: 1.7182 - val_accuracy: 0.3119

Epoch 17/1000

9727/9727 [=====] - 1s 56us/sample - loss: 2.3541 - accuracy: 0.2072 - val_loss: 1.7116 - val_accuracy: 0.3175

Epoch 18/1000

9727/9727 [=====] - 1s 54us/sample - loss: 2.3064 - accuracy: 0.2053 - val_loss: 1.7078 - val_accuracy: 0.3191

Epoch 19/1000

9727/9727 [=====] - 1s 54us/sample - loss: 2.2604 -

```
accuracy: 0.2144 - val_loss: 1.7066 - val_accuracy: 0.3184
Epoch 20/1000
9727/9727 [=====] - 1s 57us/sample - loss: 2.2109 -
accuracy: 0.2110 - val_loss: 1.7057 - val_accuracy: 0.3165
Epoch 21/1000
9727/9727 [=====] - 1s 55us/sample - loss: 2.1946 -
accuracy: 0.2132 - val_loss: 1.7036 - val_accuracy: 0.3170
Epoch 22/1000
9727/9727 [=====] - 1s 54us/sample - loss: 2.1312 -
accuracy: 0.2149 - val_loss: 1.7027 - val_accuracy: 0.3219
Epoch 23/1000
9727/9727 [=====] - 1s 54us/sample - loss: 2.1170 -
accuracy: 0.2096 - val_loss: 1.7019 - val_accuracy: 0.3205
Epoch 24/1000
9727/9727 [=====] - 1s 54us/sample - loss: 2.0889 -
accuracy: 0.2205 - val_loss: 1.7040 - val_accuracy: 0.3165
Epoch 25/1000
9727/9727 [=====] - 1s 54us/sample - loss: 2.0659 -
accuracy: 0.2165 - val_loss: 1.7039 - val_accuracy: 0.3171
Epoch 26/1000
9727/9727 [=====] - 1s 54us/sample - loss: 2.0352 -
accuracy: 0.2159 - val_loss: 1.7026 - val_accuracy: 0.3198
Epoch 27/1000
9727/9727 [=====] - 1s 54us/sample - loss: 2.0307 -
accuracy: 0.2175 - val_loss: 1.7037 - val_accuracy: 0.3185
Epoch 28/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.9999 -
accuracy: 0.2166 - val_loss: 1.7031 - val_accuracy: 0.3228
Epoch 29/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.9995 -
accuracy: 0.2113 - val_loss: 1.7040 - val_accuracy: 0.3232
Epoch 30/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.9819 -
accuracy: 0.2199 - val_loss: 1.7046 - val_accuracy: 0.3210
Epoch 31/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.9520 -
accuracy: 0.2260 - val_loss: 1.7046 - val_accuracy: 0.3218
Epoch 32/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.9479 -
accuracy: 0.2262 - val_loss: 1.7047 - val_accuracy: 0.3228
Epoch 33/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.9357 -
accuracy: 0.2295 - val_loss: 1.7055 - val_accuracy: 0.3242
Epoch 34/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.9192 -
accuracy: 0.2281 - val_loss: 1.7048 - val_accuracy: 0.3273
Epoch 35/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.9093 -
accuracy: 0.2262 - val_loss: 1.7047 - val_accuracy: 0.3253
Epoch 36/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.8998 -
accuracy: 0.2338 - val_loss: 1.7045 - val_accuracy: 0.3250
Epoch 37/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.8804 -
accuracy: 0.2352 - val_loss: 1.7028 - val_accuracy: 0.3293
Epoch 38/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.8777 -
```

```
accuracy: 0.2374 - val_loss: 1.7007 - val_accuracy: 0.3289
Epoch 39/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.8756 -
accuracy: 0.2317 - val_loss: 1.6995 - val_accuracy: 0.3336
Epoch 40/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.8709 -
accuracy: 0.2388 - val_loss: 1.6979 - val_accuracy: 0.3356
Epoch 41/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.8640 -
accuracy: 0.2369 - val_loss: 1.6982 - val_accuracy: 0.3378
Epoch 42/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.8483 -
accuracy: 0.2426 - val_loss: 1.6959 - val_accuracy: 0.3398
Epoch 43/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.8431 -
accuracy: 0.2428 - val_loss: 1.6937 - val_accuracy: 0.3426
Epoch 44/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.8393 -
accuracy: 0.2448 - val_loss: 1.6932 - val_accuracy: 0.3475
Epoch 45/1000
9727/9727 [=====] - 1s 83us/sample - loss: 1.8335 -
accuracy: 0.2426 - val_loss: 1.6920 - val_accuracy: 0.3460
Epoch 46/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.8337 -
accuracy: 0.2427 - val_loss: 1.6902 - val_accuracy: 0.3532
Epoch 47/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.8265 -
accuracy: 0.2414 - val_loss: 1.6881 - val_accuracy: 0.3575
Epoch 48/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.8083 -
accuracy: 0.2503 - val_loss: 1.6851 - val_accuracy: 0.3636
Epoch 49/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.8222 -
accuracy: 0.2512 - val_loss: 1.6845 - val_accuracy: 0.3651
Epoch 50/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.8027 -
accuracy: 0.2465 - val_loss: 1.6823 - val_accuracy: 0.3711
Epoch 51/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.8060 -
accuracy: 0.2459 - val_loss: 1.6825 - val_accuracy: 0.3697
Epoch 52/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.7971 -
accuracy: 0.2602 - val_loss: 1.6788 - val_accuracy: 0.3736
Epoch 53/1000
9727/9727 [=====] - 1s 70us/sample - loss: 1.7819 -
accuracy: 0.2632 - val_loss: 1.6754 - val_accuracy: 0.3799
Epoch 54/1000
9727/9727 [=====] - 1s 60us/sample - loss: 1.7823 -
accuracy: 0.2572 - val_loss: 1.6718 - val_accuracy: 0.3847
Epoch 55/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.7883 -
accuracy: 0.2637 - val_loss: 1.6710 - val_accuracy: 0.3868
Epoch 56/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.7816 -
accuracy: 0.2620 - val_loss: 1.6682 - val_accuracy: 0.3878
Epoch 57/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.7815 -
```

```
accuracy: 0.2601 - val_loss: 1.6660 - val_accuracy: 0.3921
Epoch 58/1000
9727/9727 [=====] - 1s 63us/sample - loss: 1.7711 -
accuracy: 0.2624 - val_loss: 1.6637 - val_accuracy: 0.3945
Epoch 59/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.7688 -
accuracy: 0.2680 - val_loss: 1.6604 - val_accuracy: 0.4002
Epoch 60/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.7708 -
accuracy: 0.2653 - val_loss: 1.6585 - val_accuracy: 0.4021
Epoch 61/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.7609 -
accuracy: 0.2694 - val_loss: 1.6550 - val_accuracy: 0.4053
Epoch 62/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.7702 -
accuracy: 0.2679 - val_loss: 1.6524 - val_accuracy: 0.4076
Epoch 63/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.7465 -
accuracy: 0.2773 - val_loss: 1.6487 - val_accuracy: 0.4124
Epoch 64/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.7535 -
accuracy: 0.2742 - val_loss: 1.6473 - val_accuracy: 0.4134
Epoch 65/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.7397 -
accuracy: 0.2829 - val_loss: 1.6432 - val_accuracy: 0.4178
Epoch 66/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.7431 -
accuracy: 0.2811 - val_loss: 1.6403 - val_accuracy: 0.4217
Epoch 67/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.7519 -
accuracy: 0.2749 - val_loss: 1.6403 - val_accuracy: 0.4218
Epoch 68/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.7333 -
accuracy: 0.2806 - val_loss: 1.6363 - val_accuracy: 0.4238
Epoch 69/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.7433 -
accuracy: 0.2759 - val_loss: 1.6334 - val_accuracy: 0.4278
Epoch 70/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.7346 -
accuracy: 0.2830 - val_loss: 1.6303 - val_accuracy: 0.4306
Epoch 71/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.7197 -
accuracy: 0.2980 - val_loss: 1.6247 - val_accuracy: 0.4352
Epoch 72/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.7190 -
accuracy: 0.2919 - val_loss: 1.6210 - val_accuracy: 0.4379
Epoch 73/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.7147 -
accuracy: 0.2942 - val_loss: 1.6161 - val_accuracy: 0.4386
Epoch 74/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.7166 -
accuracy: 0.2982 - val_loss: 1.6147 - val_accuracy: 0.4428
Epoch 75/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.7244 -
accuracy: 0.2889 - val_loss: 1.6103 - val_accuracy: 0.4442
Epoch 76/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.7128 -
```

accuracy: 0.2972 - val_loss: 1.6073 - val_accuracy: 0.4476
Epoch 77/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.7067 -
accuracy: 0.2980 - val_loss: 1.6033 - val_accuracy: 0.4499
Epoch 78/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.7077 -
accuracy: 0.2942 - val_loss: 1.6008 - val_accuracy: 0.4522
Epoch 79/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.7010 -
accuracy: 0.3007 - val_loss: 1.5970 - val_accuracy: 0.4536
Epoch 80/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.7071 -
accuracy: 0.3012 - val_loss: 1.5930 - val_accuracy: 0.4553
Epoch 81/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.6923 -
accuracy: 0.3021 - val_loss: 1.5913 - val_accuracy: 0.4553
Epoch 82/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.7036 -
accuracy: 0.2979 - val_loss: 1.5887 - val_accuracy: 0.4581
Epoch 83/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.6863 -
accuracy: 0.3070 - val_loss: 1.5848 - val_accuracy: 0.4595
Epoch 84/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.6913 -
accuracy: 0.3093 - val_loss: 1.5807 - val_accuracy: 0.4613
Epoch 85/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.6912 -
accuracy: 0.3048 - val_loss: 1.5767 - val_accuracy: 0.4616
Epoch 86/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.6809 -
accuracy: 0.3192 - val_loss: 1.5738 - val_accuracy: 0.4624
Epoch 87/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.6826 -
accuracy: 0.3182 - val_loss: 1.5700 - val_accuracy: 0.4655
Epoch 88/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.6715 -
accuracy: 0.3165 - val_loss: 1.5646 - val_accuracy: 0.4656
Epoch 89/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.6793 -
accuracy: 0.3104 - val_loss: 1.5611 - val_accuracy: 0.4664
Epoch 90/1000
9727/9727 [=====] - 1s 59us/sample - loss: 1.6655 -
accuracy: 0.3197 - val_loss: 1.5572 - val_accuracy: 0.4678
Epoch 91/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.6774 -
accuracy: 0.3140 - val_loss: 1.5542 - val_accuracy: 0.4676
Epoch 92/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.6564 -
accuracy: 0.3277 - val_loss: 1.5505 - val_accuracy: 0.4704
Epoch 93/1000
9727/9727 [=====] - 1s 61us/sample - loss: 1.6598 -
accuracy: 0.3235 - val_loss: 1.5472 - val_accuracy: 0.4692
Epoch 94/1000
9727/9727 [=====] - 1s 61us/sample - loss: 1.6574 -
accuracy: 0.3229 - val_loss: 1.5437 - val_accuracy: 0.4699
Epoch 95/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.6544 -

accuracy: 0.3296 - val_loss: 1.5395 - val_accuracy: 0.4712
Epoch 96/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.6461 -
accuracy: 0.3293 - val_loss: 1.5374 - val_accuracy: 0.4716
Epoch 97/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.6489 -
accuracy: 0.3286 - val_loss: 1.5349 - val_accuracy: 0.4732
Epoch 98/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.6460 -
accuracy: 0.3337 - val_loss: 1.5309 - val_accuracy: 0.4722
Epoch 99/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.6476 -
accuracy: 0.3295 - val_loss: 1.5267 - val_accuracy: 0.4744
Epoch 100/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.6460 -
accuracy: 0.3354 - val_loss: 1.5231 - val_accuracy: 0.4764
Epoch 101/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.6386 -
accuracy: 0.3361 - val_loss: 1.5196 - val_accuracy: 0.4756
Epoch 102/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.6461 -
accuracy: 0.3331 - val_loss: 1.5169 - val_accuracy: 0.4773
Epoch 103/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.6350 -
accuracy: 0.3379 - val_loss: 1.5125 - val_accuracy: 0.4778
Epoch 104/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.6388 -
accuracy: 0.3365 - val_loss: 1.5105 - val_accuracy: 0.4803
Epoch 105/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.6319 -
accuracy: 0.3408 - val_loss: 1.5081 - val_accuracy: 0.4820
Epoch 106/1000
9727/9727 [=====] - 1s 59us/sample - loss: 1.6263 -
accuracy: 0.3431 - val_loss: 1.5036 - val_accuracy: 0.4830
Epoch 107/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.6220 -
accuracy: 0.3435 - val_loss: 1.4985 - val_accuracy: 0.4846
Epoch 108/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.6100 -
accuracy: 0.3502 - val_loss: 1.4936 - val_accuracy: 0.4852
Epoch 109/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.6312 -
accuracy: 0.3360 - val_loss: 1.4910 - val_accuracy: 0.4867
Epoch 110/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.6177 -
accuracy: 0.3416 - val_loss: 1.4884 - val_accuracy: 0.4892
Epoch 111/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.6206 -
accuracy: 0.3480 - val_loss: 1.4850 - val_accuracy: 0.4892
Epoch 112/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.5988 -
accuracy: 0.3572 - val_loss: 1.4813 - val_accuracy: 0.4883
Epoch 113/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.6193 -
accuracy: 0.3448 - val_loss: 1.4798 - val_accuracy: 0.4889
Epoch 114/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.6009 -

accuracy: 0.3591 - val_loss: 1.4760 - val_accuracy: 0.4906
Epoch 115/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.6150 -
accuracy: 0.3519 - val_loss: 1.4742 - val_accuracy: 0.4920
Epoch 116/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.6092 -
accuracy: 0.3457 - val_loss: 1.4690 - val_accuracy: 0.4926
Epoch 117/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.6024 -
accuracy: 0.3533 - val_loss: 1.4655 - val_accuracy: 0.4938
Epoch 118/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.5829 -
accuracy: 0.3619 - val_loss: 1.4614 - val_accuracy: 0.4965
Epoch 119/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.6029 -
accuracy: 0.3605 - val_loss: 1.4585 - val_accuracy: 0.4963
Epoch 120/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.5919 -
accuracy: 0.3593 - val_loss: 1.4543 - val_accuracy: 0.4960
Epoch 121/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5949 -
accuracy: 0.3526 - val_loss: 1.4522 - val_accuracy: 0.4968
Epoch 122/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5814 -
accuracy: 0.3619 - val_loss: 1.4485 - val_accuracy: 0.4974
Epoch 123/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.5853 -
accuracy: 0.3684 - val_loss: 1.4455 - val_accuracy: 0.4980
Epoch 124/1000
9727/9727 [=====] - 1s 61us/sample - loss: 1.5886 -
accuracy: 0.3622 - val_loss: 1.4415 - val_accuracy: 0.5011
Epoch 125/1000
9727/9727 [=====] - 1s 59us/sample - loss: 1.5828 -
accuracy: 0.3621 - val_loss: 1.4389 - val_accuracy: 0.5026
Epoch 126/1000
9727/9727 [=====] - 1s 59us/sample - loss: 1.5793 -
accuracy: 0.3669 - val_loss: 1.4351 - val_accuracy: 0.5035
Epoch 127/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.5750 -
accuracy: 0.3678 - val_loss: 1.4297 - val_accuracy: 0.5039
Epoch 128/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.5660 -
accuracy: 0.3764 - val_loss: 1.4268 - val_accuracy: 0.5045
Epoch 129/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5621 -
accuracy: 0.3805 - val_loss: 1.4220 - val_accuracy: 0.5051
Epoch 130/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5727 -
accuracy: 0.3684 - val_loss: 1.4197 - val_accuracy: 0.5062
Epoch 131/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5601 -
accuracy: 0.3746 - val_loss: 1.4163 - val_accuracy: 0.5080
Epoch 132/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5643 -
accuracy: 0.3742 - val_loss: 1.4133 - val_accuracy: 0.5094
Epoch 133/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5574 -

accuracy: 0.3753 - val_loss: 1.4093 - val_accuracy: 0.5120
Epoch 134/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5498 -
accuracy: 0.3853 - val_loss: 1.4054 - val_accuracy: 0.5117
Epoch 135/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.5519 -
accuracy: 0.3799 - val_loss: 1.4013 - val_accuracy: 0.5122
Epoch 136/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5568 -
accuracy: 0.3797 - val_loss: 1.3983 - val_accuracy: 0.5125
Epoch 137/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5455 -
accuracy: 0.3830 - val_loss: 1.3948 - val_accuracy: 0.5134
Epoch 138/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5458 -
accuracy: 0.3808 - val_loss: 1.3929 - val_accuracy: 0.5139
Epoch 139/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5311 -
accuracy: 0.3906 - val_loss: 1.3888 - val_accuracy: 0.5151
Epoch 140/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5504 -
accuracy: 0.3803 - val_loss: 1.3862 - val_accuracy: 0.5151
Epoch 141/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5320 -
accuracy: 0.3878 - val_loss: 1.3836 - val_accuracy: 0.5162
Epoch 142/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5354 -
accuracy: 0.3876 - val_loss: 1.3800 - val_accuracy: 0.5165
Epoch 143/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5326 -
accuracy: 0.3937 - val_loss: 1.3776 - val_accuracy: 0.5179
Epoch 144/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.5252 -
accuracy: 0.3909 - val_loss: 1.3743 - val_accuracy: 0.5193
Epoch 145/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.5346 -
accuracy: 0.3877 - val_loss: 1.3726 - val_accuracy: 0.5204
Epoch 146/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.5208 -
accuracy: 0.3977 - val_loss: 1.3685 - val_accuracy: 0.5211
Epoch 147/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.5251 -
accuracy: 0.3922 - val_loss: 1.3658 - val_accuracy: 0.5244
Epoch 148/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.5295 -
accuracy: 0.3897 - val_loss: 1.3627 - val_accuracy: 0.5247
Epoch 149/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5106 -
accuracy: 0.4029 - val_loss: 1.3578 - val_accuracy: 0.5262
Epoch 150/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5155 -
accuracy: 0.3935 - val_loss: 1.3538 - val_accuracy: 0.5276
Epoch 151/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.5076 -
accuracy: 0.4029 - val_loss: 1.3507 - val_accuracy: 0.5284
Epoch 152/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.5173 -

```
accuracy: 0.3915 - val_loss: 1.3476 - val_accuracy: 0.5287
Epoch 153/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5070 -
accuracy: 0.4046 - val_loss: 1.3443 - val_accuracy: 0.5307
Epoch 154/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5130 -
accuracy: 0.3990 - val_loss: 1.3436 - val_accuracy: 0.5330
Epoch 155/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.5067 -
accuracy: 0.4037 - val_loss: 1.3412 - val_accuracy: 0.5328
Epoch 156/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.5086 -
accuracy: 0.4018 - val_loss: 1.3374 - val_accuracy: 0.5352
Epoch 157/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.4980 -
accuracy: 0.4056 - val_loss: 1.3355 - val_accuracy: 0.5364
Epoch 158/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.5055 -
accuracy: 0.3933 - val_loss: 1.3331 - val_accuracy: 0.5370
Epoch 159/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4883 -
accuracy: 0.4159 - val_loss: 1.3295 - val_accuracy: 0.5370
Epoch 160/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.4867 -
accuracy: 0.4109 - val_loss: 1.3264 - val_accuracy: 0.5389
Epoch 161/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4958 -
accuracy: 0.4067 - val_loss: 1.3247 - val_accuracy: 0.5389
Epoch 162/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4966 -
accuracy: 0.4108 - val_loss: 1.3233 - val_accuracy: 0.5409
Epoch 163/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4783 -
accuracy: 0.4117 - val_loss: 1.3196 - val_accuracy: 0.5422
Epoch 164/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4776 -
accuracy: 0.4160 - val_loss: 1.3160 - val_accuracy: 0.5410
Epoch 165/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.4751 -
accuracy: 0.4176 - val_loss: 1.3132 - val_accuracy: 0.5424
Epoch 166/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.4747 -
accuracy: 0.4184 - val_loss: 1.3105 - val_accuracy: 0.5456
Epoch 167/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.4711 -
accuracy: 0.4162 - val_loss: 1.3076 - val_accuracy: 0.5450
Epoch 168/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4824 -
accuracy: 0.4101 - val_loss: 1.3055 - val_accuracy: 0.5466
Epoch 169/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4726 -
accuracy: 0.4234 - val_loss: 1.3027 - val_accuracy: 0.5495
Epoch 170/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.4680 -
accuracy: 0.4189 - val_loss: 1.3001 - val_accuracy: 0.5487
Epoch 171/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4766 -
```

accuracy: 0.4177 - val_loss: 1.2970 - val_accuracy: 0.5498
Epoch 172/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4597 -
accuracy: 0.4235 - val_loss: 1.2924 - val_accuracy: 0.5521
Epoch 173/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4573 -
accuracy: 0.4240 - val_loss: 1.2889 - val_accuracy: 0.5529
Epoch 174/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4681 -
accuracy: 0.4209 - val_loss: 1.2873 - val_accuracy: 0.5538
Epoch 175/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.4561 -
accuracy: 0.4278 - val_loss: 1.2846 - val_accuracy: 0.5566
Epoch 176/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4423 -
accuracy: 0.4297 - val_loss: 1.2807 - val_accuracy: 0.5570
Epoch 177/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4535 -
accuracy: 0.4271 - val_loss: 1.2778 - val_accuracy: 0.5586
Epoch 178/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4419 -
accuracy: 0.4319 - val_loss: 1.2757 - val_accuracy: 0.5600
Epoch 179/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4410 -
accuracy: 0.4374 - val_loss: 1.2727 - val_accuracy: 0.5614
Epoch 180/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4377 -
accuracy: 0.4359 - val_loss: 1.2709 - val_accuracy: 0.5621
Epoch 181/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.4451 -
accuracy: 0.4286 - val_loss: 1.2688 - val_accuracy: 0.5629
Epoch 182/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4379 -
accuracy: 0.4335 - val_loss: 1.2655 - val_accuracy: 0.5629
Epoch 183/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.4361 -
accuracy: 0.4380 - val_loss: 1.2626 - val_accuracy: 0.5648
Epoch 184/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.4325 -
accuracy: 0.4396 - val_loss: 1.2601 - val_accuracy: 0.5654
Epoch 185/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.4435 -
accuracy: 0.4329 - val_loss: 1.2566 - val_accuracy: 0.5666
Epoch 186/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.4292 -
accuracy: 0.4394 - val_loss: 1.2526 - val_accuracy: 0.5685
Epoch 187/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.4360 -
accuracy: 0.4380 - val_loss: 1.2504 - val_accuracy: 0.5695
Epoch 188/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.4207 -
accuracy: 0.4378 - val_loss: 1.2467 - val_accuracy: 0.5705
Epoch 189/1000
9727/9727 [=====] - 1s 60us/sample - loss: 1.4223 -
accuracy: 0.4452 - val_loss: 1.2452 - val_accuracy: 0.5703
Epoch 190/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.4223 -

accuracy: 0.4456 - val_loss: 1.2422 - val_accuracy: 0.5732
Epoch 191/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.4276 -
accuracy: 0.4433 - val_loss: 1.2391 - val_accuracy: 0.5735
Epoch 192/1000
9727/9727 [=====] - 1s 62us/sample - loss: 1.4211 -
accuracy: 0.4429 - val_loss: 1.2377 - val_accuracy: 0.5748
Epoch 193/1000
9727/9727 [=====] - 1s 64us/sample - loss: 1.4121 -
accuracy: 0.4476 - val_loss: 1.2352 - val_accuracy: 0.5751
Epoch 194/1000
9727/9727 [=====] - 1s 71us/sample - loss: 1.4116 -
accuracy: 0.4481 - val_loss: 1.2324 - val_accuracy: 0.5768
Epoch 195/1000
9727/9727 [=====] - 1s 63us/sample - loss: 1.4015 -
accuracy: 0.4529 - val_loss: 1.2300 - val_accuracy: 0.5789
Epoch 196/1000
9727/9727 [=====] - 1s 64us/sample - loss: 1.4178 -
accuracy: 0.4455 - val_loss: 1.2270 - val_accuracy: 0.5791
Epoch 197/1000
9727/9727 [=====] - 1s 60us/sample - loss: 1.4085 -
accuracy: 0.4552 - val_loss: 1.2239 - val_accuracy: 0.5816
Epoch 198/1000
9727/9727 [=====] - 1s 60us/sample - loss: 1.4120 -
accuracy: 0.4474 - val_loss: 1.2210 - val_accuracy: 0.5811
Epoch 199/1000
9727/9727 [=====] - 1s 60us/sample - loss: 1.4004 -
accuracy: 0.4580 - val_loss: 1.2193 - val_accuracy: 0.5814
Epoch 200/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.3914 -
accuracy: 0.4595 - val_loss: 1.2168 - val_accuracy: 0.5825
Epoch 201/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.4048 -
accuracy: 0.4506 - val_loss: 1.2146 - val_accuracy: 0.5848
Epoch 202/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.4017 -
accuracy: 0.4531 - val_loss: 1.2124 - val_accuracy: 0.5865
Epoch 203/1000
9727/9727 [=====] - 1s 64us/sample - loss: 1.3969 -
accuracy: 0.4518 - val_loss: 1.2097 - val_accuracy: 0.5863
Epoch 204/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.3872 -
accuracy: 0.4567 - val_loss: 1.2058 - val_accuracy: 0.5876
Epoch 205/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3873 -
accuracy: 0.4566 - val_loss: 1.2035 - val_accuracy: 0.5890
Epoch 206/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.3962 -
accuracy: 0.4613 - val_loss: 1.2013 - val_accuracy: 0.5899
Epoch 207/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3873 -
accuracy: 0.4629 - val_loss: 1.1983 - val_accuracy: 0.5903
Epoch 208/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3908 -
accuracy: 0.4598 - val_loss: 1.1968 - val_accuracy: 0.5916
Epoch 209/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.3877 -

```
accuracy: 0.4616 - val_loss: 1.1948 - val_accuracy: 0.5934
Epoch 210/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.3851 -
accuracy: 0.4566 - val_loss: 1.1921 - val_accuracy: 0.5930
Epoch 211/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.3902 -
accuracy: 0.4546 - val_loss: 1.1895 - val_accuracy: 0.5945
Epoch 212/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.3754 -
accuracy: 0.4683 - val_loss: 1.1871 - val_accuracy: 0.5950
Epoch 213/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.3714 -
accuracy: 0.4700 - val_loss: 1.1855 - val_accuracy: 0.5976
Epoch 214/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3717 -
accuracy: 0.4658 - val_loss: 1.1833 - val_accuracy: 0.5982
Epoch 215/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3626 -
accuracy: 0.4708 - val_loss: 1.1803 - val_accuracy: 0.6001
Epoch 216/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.3720 -
accuracy: 0.4650 - val_loss: 1.1779 - val_accuracy: 0.6004
Epoch 217/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.3632 -
accuracy: 0.4751 - val_loss: 1.1759 - val_accuracy: 0.6013
Epoch 218/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.3643 -
accuracy: 0.4720 - val_loss: 1.1746 - val_accuracy: 0.6021
Epoch 219/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3677 -
accuracy: 0.4704 - val_loss: 1.1733 - val_accuracy: 0.6036
Epoch 220/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.3694 -
accuracy: 0.4679 - val_loss: 1.1710 - val_accuracy: 0.6035
Epoch 221/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.3663 -
accuracy: 0.4674 - val_loss: 1.1694 - val_accuracy: 0.6048
Epoch 222/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.3662 -
accuracy: 0.4718 - val_loss: 1.1683 - val_accuracy: 0.6056
Epoch 223/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.3643 -
accuracy: 0.4725 - val_loss: 1.1674 - val_accuracy: 0.6067
Epoch 224/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3622 -
accuracy: 0.4699 - val_loss: 1.1655 - val_accuracy: 0.6090
Epoch 225/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.3498 -
accuracy: 0.4776 - val_loss: 1.1627 - val_accuracy: 0.6082
Epoch 226/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.3336 -
accuracy: 0.4870 - val_loss: 1.1586 - val_accuracy: 0.6102
Epoch 227/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.3526 -
accuracy: 0.4730 - val_loss: 1.1554 - val_accuracy: 0.6109
Epoch 228/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.3511 -
```

accuracy: 0.4761 - val_loss: 1.1527 - val_accuracy: 0.6115
Epoch 229/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3426 -
accuracy: 0.4800 - val_loss: 1.1504 - val_accuracy: 0.6112
Epoch 230/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.3492 -
accuracy: 0.4737 - val_loss: 1.1489 - val_accuracy: 0.6126
Epoch 231/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3395 -
accuracy: 0.4799 - val_loss: 1.1469 - val_accuracy: 0.6110
Epoch 232/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.3304 -
accuracy: 0.4882 - val_loss: 1.1445 - val_accuracy: 0.6127
Epoch 233/1000
9727/9727 [=====] - 1s 60us/sample - loss: 1.3229 -
accuracy: 0.4885 - val_loss: 1.1420 - val_accuracy: 0.6149
Epoch 234/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.3409 -
accuracy: 0.4786 - val_loss: 1.1409 - val_accuracy: 0.6146
Epoch 235/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3428 -
accuracy: 0.4848 - val_loss: 1.1388 - val_accuracy: 0.6161
Epoch 236/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.3291 -
accuracy: 0.4913 - val_loss: 1.1364 - val_accuracy: 0.6172
Epoch 237/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.3490 -
accuracy: 0.4763 - val_loss: 1.1366 - val_accuracy: 0.6176
Epoch 238/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.3209 -
accuracy: 0.4842 - val_loss: 1.1328 - val_accuracy: 0.6193
Epoch 239/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3186 -
accuracy: 0.4893 - val_loss: 1.1300 - val_accuracy: 0.6179
Epoch 240/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3367 -
accuracy: 0.4840 - val_loss: 1.1280 - val_accuracy: 0.6227
Epoch 241/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3180 -
accuracy: 0.4956 - val_loss: 1.1253 - val_accuracy: 0.6241
Epoch 242/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.3283 -
accuracy: 0.4911 - val_loss: 1.1236 - val_accuracy: 0.6253
Epoch 243/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3098 -
accuracy: 0.4972 - val_loss: 1.1218 - val_accuracy: 0.6278
Epoch 244/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.3257 -
accuracy: 0.4892 - val_loss: 1.1206 - val_accuracy: 0.6252
Epoch 245/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3058 -
accuracy: 0.4995 - val_loss: 1.1176 - val_accuracy: 0.6275
Epoch 246/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3240 -
accuracy: 0.4883 - val_loss: 1.1162 - val_accuracy: 0.6283
Epoch 247/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.3139 -

accuracy: 0.4952 - val_loss: 1.1140 - val_accuracy: 0.6287
Epoch 248/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3153 -
accuracy: 0.4920 - val_loss: 1.1121 - val_accuracy: 0.6307
Epoch 249/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3087 -
accuracy: 0.4993 - val_loss: 1.1094 - val_accuracy: 0.6309
Epoch 250/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.3093 -
accuracy: 0.5023 - val_loss: 1.1069 - val_accuracy: 0.6318
Epoch 251/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2940 -
accuracy: 0.5012 - val_loss: 1.1048 - val_accuracy: 0.6317
Epoch 252/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.3130 -
accuracy: 0.5038 - val_loss: 1.1044 - val_accuracy: 0.6332
Epoch 253/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2921 -
accuracy: 0.5074 - val_loss: 1.1020 - val_accuracy: 0.6343
Epoch 254/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.3171 -
accuracy: 0.4956 - val_loss: 1.1010 - val_accuracy: 0.6349
Epoch 255/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.2967 -
accuracy: 0.4996 - val_loss: 1.0983 - val_accuracy: 0.6358
Epoch 256/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.2878 -
accuracy: 0.5060 - val_loss: 1.0969 - val_accuracy: 0.6363
Epoch 257/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.2915 -
accuracy: 0.5055 - val_loss: 1.0949 - val_accuracy: 0.6352
Epoch 258/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.2750 -
accuracy: 0.5134 - val_loss: 1.0910 - val_accuracy: 0.6377
Epoch 259/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.2952 -
accuracy: 0.5030 - val_loss: 1.0894 - val_accuracy: 0.6391
Epoch 260/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.2919 -
accuracy: 0.5022 - val_loss: 1.0877 - val_accuracy: 0.6403
Epoch 261/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.2887 -
accuracy: 0.5100 - val_loss: 1.0868 - val_accuracy: 0.6406
Epoch 262/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.2948 -
accuracy: 0.5088 - val_loss: 1.0851 - val_accuracy: 0.6422
Epoch 263/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2860 -
accuracy: 0.5089 - val_loss: 1.0820 - val_accuracy: 0.6426
Epoch 264/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2822 -
accuracy: 0.5111 - val_loss: 1.0809 - val_accuracy: 0.6432
Epoch 265/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2867 -
accuracy: 0.5064 - val_loss: 1.0798 - val_accuracy: 0.6429
Epoch 266/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2855 -


```
accuracy: 0.5085 - val_loss: 1.0787 - val_accuracy: 0.6442
Epoch 267/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2819 -
accuracy: 0.5086 - val_loss: 1.0768 - val_accuracy: 0.6442
Epoch 268/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2790 -
accuracy: 0.5120 - val_loss: 1.0746 - val_accuracy: 0.6448
Epoch 269/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.2757 -
accuracy: 0.5066 - val_loss: 1.0736 - val_accuracy: 0.6454
Epoch 270/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2757 -
accuracy: 0.5134 - val_loss: 1.0725 - val_accuracy: 0.6448
Epoch 271/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2758 -
accuracy: 0.5094 - val_loss: 1.0708 - val_accuracy: 0.6468
Epoch 272/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.2644 -
accuracy: 0.5203 - val_loss: 1.0686 - val_accuracy: 0.6469
Epoch 273/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2681 -
accuracy: 0.5160 - val_loss: 1.0666 - val_accuracy: 0.6477
Epoch 274/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2555 -
accuracy: 0.5237 - val_loss: 1.0635 - val_accuracy: 0.6485
Epoch 275/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2633 -
accuracy: 0.5261 - val_loss: 1.0622 - val_accuracy: 0.6489
Epoch 276/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2707 -
accuracy: 0.5087 - val_loss: 1.0608 - val_accuracy: 0.6499
Epoch 277/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.2568 -
accuracy: 0.5176 - val_loss: 1.0588 - val_accuracy: 0.6502
Epoch 278/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2494 -
accuracy: 0.5266 - val_loss: 1.0564 - val_accuracy: 0.6509
Epoch 279/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2545 -
accuracy: 0.5254 - val_loss: 1.0550 - val_accuracy: 0.6520
Epoch 280/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2504 -
accuracy: 0.5285 - val_loss: 1.0535 - val_accuracy: 0.6533
Epoch 281/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2554 -
accuracy: 0.5167 - val_loss: 1.0519 - val_accuracy: 0.6528
Epoch 282/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.2442 -
accuracy: 0.5312 - val_loss: 1.0499 - val_accuracy: 0.6533
Epoch 283/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2460 -
accuracy: 0.5192 - val_loss: 1.0481 - val_accuracy: 0.6545
Epoch 284/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2510 -
accuracy: 0.5266 - val_loss: 1.0464 - val_accuracy: 0.6533
Epoch 285/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2455 -
```

```
accuracy: 0.5261 - val_loss: 1.0448 - val_accuracy: 0.6548
Epoch 286/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2522 -
accuracy: 0.5249 - val_loss: 1.0437 - val_accuracy: 0.6560
Epoch 287/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2427 -
accuracy: 0.5228 - val_loss: 1.0428 - val_accuracy: 0.6562
Epoch 288/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2416 -
accuracy: 0.5271 - val_loss: 1.0416 - val_accuracy: 0.6576
Epoch 289/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2459 -
accuracy: 0.5280 - val_loss: 1.0402 - val_accuracy: 0.6571
Epoch 290/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2337 -
accuracy: 0.5315 - val_loss: 1.0379 - val_accuracy: 0.6580
Epoch 291/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.2433 -
accuracy: 0.5274 - val_loss: 1.0359 - val_accuracy: 0.6580
Epoch 292/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.2359 -
accuracy: 0.5363 - val_loss: 1.0348 - val_accuracy: 0.6593
Epoch 293/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2331 -
accuracy: 0.5311 - val_loss: 1.0329 - val_accuracy: 0.6586
Epoch 294/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.2394 -
accuracy: 0.5288 - val_loss: 1.0319 - val_accuracy: 0.6600
Epoch 295/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2463 -
accuracy: 0.5278 - val_loss: 1.0304 - val_accuracy: 0.6605
Epoch 296/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.2394 -
accuracy: 0.5311 - val_loss: 1.0292 - val_accuracy: 0.6616
Epoch 297/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2191 -
accuracy: 0.5394 - val_loss: 1.0267 - val_accuracy: 0.6608
Epoch 298/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2310 -
accuracy: 0.5367 - val_loss: 1.0251 - val_accuracy: 0.6610
Epoch 299/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2286 -
accuracy: 0.5415 - val_loss: 1.0239 - val_accuracy: 0.6617
Epoch 300/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.2342 -
accuracy: 0.5336 - val_loss: 1.0238 - val_accuracy: 0.6627
Epoch 301/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.2330 -
accuracy: 0.5335 - val_loss: 1.0215 - val_accuracy: 0.6631
Epoch 302/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.2326 -
accuracy: 0.5349 - val_loss: 1.0214 - val_accuracy: 0.6644
Epoch 303/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.2279 -
accuracy: 0.5300 - val_loss: 1.0203 - val_accuracy: 0.6640
Epoch 304/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.2217 -
```

accuracy: 0.5385 - val_loss: 1.0187 - val_accuracy: 0.6651
Epoch 305/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.2239 -
accuracy: 0.5428 - val_loss: 1.0175 - val_accuracy: 0.6656
Epoch 306/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2114 -
accuracy: 0.5481 - val_loss: 1.0165 - val_accuracy: 0.6660
Epoch 307/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.2042 -
accuracy: 0.5401 - val_loss: 1.0143 - val_accuracy: 0.6660
Epoch 308/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.1995 -
accuracy: 0.5454 - val_loss: 1.0117 - val_accuracy: 0.6654
Epoch 309/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.2085 -
accuracy: 0.5429 - val_loss: 1.0091 - val_accuracy: 0.6671
Epoch 310/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.1993 -
accuracy: 0.5474 - val_loss: 1.0068 - val_accuracy: 0.6676
Epoch 311/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.2084 -
accuracy: 0.5445 - val_loss: 1.0064 - val_accuracy: 0.6677
Epoch 312/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.2067 -
accuracy: 0.5439 - val_loss: 1.0052 - val_accuracy: 0.6691
Epoch 313/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.2123 -
accuracy: 0.5478 - val_loss: 1.0047 - val_accuracy: 0.6698
Epoch 314/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.2037 -
accuracy: 0.5488 - val_loss: 1.0033 - val_accuracy: 0.6694
Epoch 315/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.2063 -
accuracy: 0.5471 - val_loss: 1.0015 - val_accuracy: 0.6694
Epoch 316/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.2090 -
accuracy: 0.5479 - val_loss: 1.0003 - val_accuracy: 0.6699
Epoch 317/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.1994 -
accuracy: 0.5517 - val_loss: 0.9995 - val_accuracy: 0.6704
Epoch 318/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.2067 -
accuracy: 0.5462 - val_loss: 0.9982 - val_accuracy: 0.6708
Epoch 319/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.2053 -
accuracy: 0.5507 - val_loss: 0.9966 - val_accuracy: 0.6721
Epoch 320/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.2024 -
accuracy: 0.5456 - val_loss: 0.9957 - val_accuracy: 0.6727
Epoch 321/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.1917 -
accuracy: 0.5482 - val_loss: 0.9930 - val_accuracy: 0.6728
Epoch 322/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.1899 -
accuracy: 0.5548 - val_loss: 0.9913 - val_accuracy: 0.6733
Epoch 323/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1910 -

accuracy: 0.5463 - val_loss: 0.9906 - val_accuracy: 0.6733
Epoch 324/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.1963 -
accuracy: 0.5544 - val_loss: 0.9895 - val_accuracy: 0.6730
Epoch 325/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.1950 -
accuracy: 0.5461 - val_loss: 0.9884 - val_accuracy: 0.6739
Epoch 326/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.1888 -
accuracy: 0.5553 - val_loss: 0.9876 - val_accuracy: 0.6741
Epoch 327/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.1827 -
accuracy: 0.5535 - val_loss: 0.9855 - val_accuracy: 0.6751
Epoch 328/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.1824 -
accuracy: 0.5579 - val_loss: 0.9844 - val_accuracy: 0.6748
Epoch 329/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1749 -
accuracy: 0.5610 - val_loss: 0.9831 - val_accuracy: 0.6742
Epoch 330/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.1811 -
accuracy: 0.5594 - val_loss: 0.9818 - val_accuracy: 0.6751
Epoch 331/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.1841 -
accuracy: 0.5571 - val_loss: 0.9805 - val_accuracy: 0.6758
Epoch 332/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.1865 -
accuracy: 0.5603 - val_loss: 0.9787 - val_accuracy: 0.6753
Epoch 333/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.1968 -
accuracy: 0.5572 - val_loss: 0.9784 - val_accuracy: 0.6767
Epoch 334/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1865 -
accuracy: 0.5587 - val_loss: 0.9767 - val_accuracy: 0.6773
Epoch 335/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.1773 -
accuracy: 0.5538 - val_loss: 0.9752 - val_accuracy: 0.6781
Epoch 336/1000
9727/9727 [=====] - 1s 62us/sample - loss: 1.1727 -
accuracy: 0.5595 - val_loss: 0.9742 - val_accuracy: 0.6779
Epoch 337/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.1840 -
accuracy: 0.5574 - val_loss: 0.9734 - val_accuracy: 0.6782
Epoch 338/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.1931 -
accuracy: 0.5531 - val_loss: 0.9727 - val_accuracy: 0.6795
Epoch 339/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.1749 -
accuracy: 0.5604 - val_loss: 0.9718 - val_accuracy: 0.6790
Epoch 340/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1818 -
accuracy: 0.5617 - val_loss: 0.9713 - val_accuracy: 0.6788
Epoch 341/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.1688 -
accuracy: 0.5618 - val_loss: 0.9695 - val_accuracy: 0.6790
Epoch 342/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.1720 -

accuracy: 0.5601 - val_loss: 0.9691 - val_accuracy: 0.6802
Epoch 343/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.1681 -
accuracy: 0.5632 - val_loss: 0.9673 - val_accuracy: 0.6813
Epoch 344/1000
9727/9727 [=====] - 1s 59us/sample - loss: 1.1698 -
accuracy: 0.5642 - val_loss: 0.9659 - val_accuracy: 0.6830
Epoch 345/1000
9727/9727 [=====] - 1s 60us/sample - loss: 1.1656 -
accuracy: 0.5691 - val_loss: 0.9643 - val_accuracy: 0.6830
Epoch 346/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1656 -
accuracy: 0.5689 - val_loss: 0.9635 - val_accuracy: 0.6830
Epoch 347/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.1583 -
accuracy: 0.5602 - val_loss: 0.9621 - val_accuracy: 0.6836
Epoch 348/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.1656 -
accuracy: 0.5676 - val_loss: 0.9602 - val_accuracy: 0.6836
Epoch 349/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1615 -
accuracy: 0.5676 - val_loss: 0.9594 - val_accuracy: 0.6844
Epoch 350/1000
9727/9727 [=====] - 1s 62us/sample - loss: 1.1584 -
accuracy: 0.5697 - val_loss: 0.9586 - val_accuracy: 0.6839
Epoch 351/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1645 -
accuracy: 0.5704 - val_loss: 0.9584 - val_accuracy: 0.6844
Epoch 352/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1557 -
accuracy: 0.5709 - val_loss: 0.9569 - val_accuracy: 0.6852
Epoch 353/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.1531 -
accuracy: 0.5681 - val_loss: 0.9553 - val_accuracy: 0.6853
Epoch 354/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.1508 -
accuracy: 0.5703 - val_loss: 0.9544 - val_accuracy: 0.6847
Epoch 355/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1532 -
accuracy: 0.5673 - val_loss: 0.9533 - val_accuracy: 0.6862
Epoch 356/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1527 -
accuracy: 0.5684 - val_loss: 0.9518 - val_accuracy: 0.6870
Epoch 357/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1441 -
accuracy: 0.5772 - val_loss: 0.9510 - val_accuracy: 0.6861
Epoch 358/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1467 -
accuracy: 0.5755 - val_loss: 0.9496 - val_accuracy: 0.6870
Epoch 359/1000
9727/9727 [=====] - 1s 61us/sample - loss: 1.1449 -
accuracy: 0.5757 - val_loss: 0.9481 - val_accuracy: 0.6870
Epoch 360/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1636 -
accuracy: 0.5627 - val_loss: 0.9477 - val_accuracy: 0.6881
Epoch 361/1000
9727/9727 [=====] - 1s 59us/sample - loss: 1.1444 -

accuracy: 0.5723 - val_loss: 0.9465 - val_accuracy: 0.6884
Epoch 362/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1466 -
accuracy: 0.5654 - val_loss: 0.9455 - val_accuracy: 0.6892
Epoch 363/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1450 -
accuracy: 0.5747 - val_loss: 0.9445 - val_accuracy: 0.6883
Epoch 364/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1382 -
accuracy: 0.5779 - val_loss: 0.9439 - val_accuracy: 0.6895
Epoch 365/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1456 -
accuracy: 0.5790 - val_loss: 0.9420 - val_accuracy: 0.6883
Epoch 366/1000
9727/9727 [=====] - 1s 59us/sample - loss: 1.1418 -
accuracy: 0.5719 - val_loss: 0.9407 - val_accuracy: 0.6889
Epoch 367/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1417 -
accuracy: 0.5774 - val_loss: 0.9394 - val_accuracy: 0.6890
Epoch 368/1000
9727/9727 [=====] - 1s 63us/sample - loss: 1.1333 -
accuracy: 0.5780 - val_loss: 0.9385 - val_accuracy: 0.6892
Epoch 369/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1313 -
accuracy: 0.5872 - val_loss: 0.9376 - val_accuracy: 0.6889
Epoch 370/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1428 -
accuracy: 0.5761 - val_loss: 0.9366 - val_accuracy: 0.6887
Epoch 371/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1376 -
accuracy: 0.5756 - val_loss: 0.9351 - val_accuracy: 0.6889
Epoch 372/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1281 -
accuracy: 0.5853 - val_loss: 0.9344 - val_accuracy: 0.6901
Epoch 373/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1322 -
accuracy: 0.5790 - val_loss: 0.9330 - val_accuracy: 0.6909
Epoch 374/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1309 -
accuracy: 0.5832 - val_loss: 0.9320 - val_accuracy: 0.6924
Epoch 375/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1343 -
accuracy: 0.5830 - val_loss: 0.9315 - val_accuracy: 0.6936
Epoch 376/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1275 -
accuracy: 0.5773 - val_loss: 0.9317 - val_accuracy: 0.6940
Epoch 377/1000
9727/9727 [=====] - 1s 61us/sample - loss: 1.1216 -
accuracy: 0.5845 - val_loss: 0.9297 - val_accuracy: 0.6938
Epoch 378/1000
9727/9727 [=====] - 1s 59us/sample - loss: 1.1182 -
accuracy: 0.5864 - val_loss: 0.9283 - val_accuracy: 0.6935
Epoch 379/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1252 -
accuracy: 0.5868 - val_loss: 0.9268 - val_accuracy: 0.6929
Epoch 380/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1093 -

accuracy: 0.5950 - val_loss: 0.9255 - val_accuracy: 0.6940
Epoch 381/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1224 -
accuracy: 0.5804 - val_loss: 0.9244 - val_accuracy: 0.6957
Epoch 382/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1310 -
accuracy: 0.5834 - val_loss: 0.9237 - val_accuracy: 0.6955
Epoch 383/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1184 -
accuracy: 0.5866 - val_loss: 0.9224 - val_accuracy: 0.6960
Epoch 384/1000
9727/9727 [=====] - 1s 59us/sample - loss: 1.1230 -
accuracy: 0.5873 - val_loss: 0.9218 - val_accuracy: 0.6946
Epoch 385/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1242 -
accuracy: 0.5852 - val_loss: 0.9209 - val_accuracy: 0.6949
Epoch 386/1000
9727/9727 [=====] - 1s 59us/sample - loss: 1.1148 -
accuracy: 0.5900 - val_loss: 0.9203 - val_accuracy: 0.6949
Epoch 387/1000
9727/9727 [=====] - 1s 60us/sample - loss: 1.1144 -
accuracy: 0.5906 - val_loss: 0.9188 - val_accuracy: 0.6961
Epoch 388/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1175 -
accuracy: 0.5892 - val_loss: 0.9175 - val_accuracy: 0.6961
Epoch 389/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.1056 -
accuracy: 0.5915 - val_loss: 0.9161 - val_accuracy: 0.6969
Epoch 390/1000
9727/9727 [=====] - 1s 59us/sample - loss: 1.1213 -
accuracy: 0.5840 - val_loss: 0.9152 - val_accuracy: 0.6975
Epoch 391/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.1102 -
accuracy: 0.5963 - val_loss: 0.9141 - val_accuracy: 0.6958
Epoch 392/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.1117 -
accuracy: 0.5903 - val_loss: 0.9133 - val_accuracy: 0.6972
Epoch 393/1000
9727/9727 [=====] - 1s 61us/sample - loss: 1.1075 -
accuracy: 0.5909 - val_loss: 0.9124 - val_accuracy: 0.6966
Epoch 394/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.1148 -
accuracy: 0.5963 - val_loss: 0.9122 - val_accuracy: 0.6978
Epoch 395/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.1116 -
accuracy: 0.5876 - val_loss: 0.9115 - val_accuracy: 0.6975
Epoch 396/1000
9727/9727 [=====] - 1s 59us/sample - loss: 1.1011 -
accuracy: 0.5945 - val_loss: 0.9103 - val_accuracy: 0.6978
Epoch 397/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.0976 -
accuracy: 0.5942 - val_loss: 0.9093 - val_accuracy: 0.6984
Epoch 398/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.0972 -
accuracy: 0.5973 - val_loss: 0.9076 - val_accuracy: 0.6987
Epoch 399/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.1101 -

accuracy: 0.5848 - val_loss: 0.9063 - val_accuracy: 0.6980
Epoch 400/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.0882 -
accuracy: 0.5981 - val_loss: 0.9041 - val_accuracy: 0.6984
Epoch 401/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.1041 -
accuracy: 0.5904 - val_loss: 0.9034 - val_accuracy: 0.7006
Epoch 402/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.1017 -
accuracy: 0.5932 - val_loss: 0.9023 - val_accuracy: 0.7014
Epoch 403/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0990 -
accuracy: 0.5950 - val_loss: 0.9014 - val_accuracy: 0.7007
Epoch 404/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0864 -
accuracy: 0.6015 - val_loss: 0.9002 - val_accuracy: 0.7026
Epoch 405/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0924 -
accuracy: 0.6020 - val_loss: 0.8998 - val_accuracy: 0.7026
Epoch 406/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.0979 -
accuracy: 0.5940 - val_loss: 0.8987 - val_accuracy: 0.7031
Epoch 407/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0976 -
accuracy: 0.5923 - val_loss: 0.8981 - val_accuracy: 0.7034
Epoch 408/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.1005 -
accuracy: 0.5885 - val_loss: 0.8977 - val_accuracy: 0.7037
Epoch 409/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0853 -
accuracy: 0.5979 - val_loss: 0.8965 - val_accuracy: 0.7041
Epoch 410/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0946 -
accuracy: 0.5999 - val_loss: 0.8955 - val_accuracy: 0.7052
Epoch 411/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0913 -
accuracy: 0.5932 - val_loss: 0.8953 - val_accuracy: 0.7057
Epoch 412/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0960 -
accuracy: 0.5957 - val_loss: 0.8944 - val_accuracy: 0.7057
Epoch 413/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0780 -
accuracy: 0.6091 - val_loss: 0.8936 - val_accuracy: 0.7063
Epoch 414/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0832 -
accuracy: 0.5988 - val_loss: 0.8929 - val_accuracy: 0.7052
Epoch 415/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0940 -
accuracy: 0.5955 - val_loss: 0.8923 - val_accuracy: 0.7068
Epoch 416/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.0852 -
accuracy: 0.6018 - val_loss: 0.8908 - val_accuracy: 0.7068
Epoch 417/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0977 -
accuracy: 0.5950 - val_loss: 0.8904 - val_accuracy: 0.7080
Epoch 418/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0742 -

accuracy: 0.6075 - val_loss: 0.8889 - val_accuracy: 0.7078
Epoch 419/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0732 -
accuracy: 0.6055 - val_loss: 0.8878 - val_accuracy: 0.7080
Epoch 420/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0839 -
accuracy: 0.5998 - val_loss: 0.8870 - val_accuracy: 0.7089
Epoch 421/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0765 -
accuracy: 0.6043 - val_loss: 0.8860 - val_accuracy: 0.7072
Epoch 422/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0635 -
accuracy: 0.6121 - val_loss: 0.8846 - val_accuracy: 0.7088
Epoch 423/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0722 -
accuracy: 0.6091 - val_loss: 0.8837 - val_accuracy: 0.7091
Epoch 424/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0854 -
accuracy: 0.6029 - val_loss: 0.8830 - val_accuracy: 0.7094
Epoch 425/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.0720 -
accuracy: 0.6052 - val_loss: 0.8820 - val_accuracy: 0.7088
Epoch 426/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0668 -
accuracy: 0.6070 - val_loss: 0.8812 - val_accuracy: 0.7105
Epoch 427/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0825 -
accuracy: 0.6037 - val_loss: 0.8802 - val_accuracy: 0.7106
Epoch 428/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0788 -
accuracy: 0.6056 - val_loss: 0.8789 - val_accuracy: 0.7112
Epoch 429/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0672 -
accuracy: 0.6093 - val_loss: 0.8777 - val_accuracy: 0.7117
Epoch 430/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0768 -
accuracy: 0.6113 - val_loss: 0.8773 - val_accuracy: 0.7115
Epoch 431/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0710 -
accuracy: 0.6045 - val_loss: 0.8765 - val_accuracy: 0.7115
Epoch 432/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0623 -
accuracy: 0.6095 - val_loss: 0.8757 - val_accuracy: 0.7120
Epoch 433/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0819 -
accuracy: 0.6036 - val_loss: 0.8757 - val_accuracy: 0.7134
Epoch 434/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0745 -
accuracy: 0.6072 - val_loss: 0.8746 - val_accuracy: 0.7128
Epoch 435/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.0711 -
accuracy: 0.6046 - val_loss: 0.8733 - val_accuracy: 0.7125
Epoch 436/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0571 -
accuracy: 0.6150 - val_loss: 0.8724 - val_accuracy: 0.7131
Epoch 437/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0665 -

```
accuracy: 0.6079 - val_loss: 0.8719 - val_accuracy: 0.7126
Epoch 438/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.0665 -
accuracy: 0.6060 - val_loss: 0.8717 - val_accuracy: 0.7128
Epoch 439/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.0633 -
accuracy: 0.6138 - val_loss: 0.8713 - val_accuracy: 0.7112
Epoch 440/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0685 -
accuracy: 0.6098 - val_loss: 0.8711 - val_accuracy: 0.7115
Epoch 441/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0482 -
accuracy: 0.6153 - val_loss: 0.8698 - val_accuracy: 0.7125
Epoch 442/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0554 -
accuracy: 0.6150 - val_loss: 0.8690 - val_accuracy: 0.7135
Epoch 443/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0588 -
accuracy: 0.6108 - val_loss: 0.8683 - val_accuracy: 0.7135
Epoch 444/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0730 -
accuracy: 0.6103 - val_loss: 0.8676 - val_accuracy: 0.7131
Epoch 445/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.0647 -
accuracy: 0.6067 - val_loss: 0.8672 - val_accuracy: 0.7126
Epoch 446/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0687 -
accuracy: 0.6073 - val_loss: 0.8671 - val_accuracy: 0.7128
Epoch 447/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0480 -
accuracy: 0.6184 - val_loss: 0.8652 - val_accuracy: 0.7137
Epoch 448/1000
9727/9727 [=====] - 1s 53us/sample - loss: 1.0477 -
accuracy: 0.6179 - val_loss: 0.8644 - val_accuracy: 0.7151
Epoch 449/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0601 -
accuracy: 0.6150 - val_loss: 0.8635 - val_accuracy: 0.7155
Epoch 450/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0514 -
accuracy: 0.6179 - val_loss: 0.8633 - val_accuracy: 0.7145
Epoch 451/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0446 -
accuracy: 0.6197 - val_loss: 0.8625 - val_accuracy: 0.7142
Epoch 452/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0480 -
accuracy: 0.6187 - val_loss: 0.8619 - val_accuracy: 0.7149
Epoch 453/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0447 -
accuracy: 0.6188 - val_loss: 0.8608 - val_accuracy: 0.7146
Epoch 454/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0521 -
accuracy: 0.6204 - val_loss: 0.8594 - val_accuracy: 0.7157
Epoch 455/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.0618 -
accuracy: 0.6133 - val_loss: 0.8592 - val_accuracy: 0.7154
Epoch 456/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0458 -
```

```
accuracy: 0.6233 - val_loss: 0.8582 - val_accuracy: 0.7155
Epoch 457/1000
9727/9727 [=====] - 1s 61us/sample - loss: 1.0480 -
accuracy: 0.6205 - val_loss: 0.8574 - val_accuracy: 0.7154
Epoch 458/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.0505 -
accuracy: 0.6139 - val_loss: 0.8566 - val_accuracy: 0.7160
Epoch 459/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.0515 -
accuracy: 0.6107 - val_loss: 0.8559 - val_accuracy: 0.7168
Epoch 460/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.0447 -
accuracy: 0.6182 - val_loss: 0.8551 - val_accuracy: 0.7177
Epoch 461/1000
9727/9727 [=====] - 1s 60us/sample - loss: 1.0442 -
accuracy: 0.6116 - val_loss: 0.8545 - val_accuracy: 0.7182
Epoch 462/1000
9727/9727 [=====] - 1s 61us/sample - loss: 1.0524 -
accuracy: 0.6111 - val_loss: 0.8543 - val_accuracy: 0.7177
Epoch 463/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.0506 -
accuracy: 0.6207 - val_loss: 0.8537 - val_accuracy: 0.7185
Epoch 464/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.0377 -
accuracy: 0.6213 - val_loss: 0.8526 - val_accuracy: 0.7182
Epoch 465/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.0366 -
accuracy: 0.6169 - val_loss: 0.8520 - val_accuracy: 0.7180
Epoch 466/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.0402 -
accuracy: 0.6213 - val_loss: 0.8509 - val_accuracy: 0.7186
Epoch 467/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0340 -
accuracy: 0.6199 - val_loss: 0.8496 - val_accuracy: 0.7183
Epoch 468/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0352 -
accuracy: 0.6237 - val_loss: 0.8488 - val_accuracy: 0.7186
Epoch 469/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.0361 -
accuracy: 0.6210 - val_loss: 0.8477 - val_accuracy: 0.7199
Epoch 470/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.0213 -
accuracy: 0.6246 - val_loss: 0.8466 - val_accuracy: 0.7192
Epoch 471/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.0352 -
accuracy: 0.6269 - val_loss: 0.8456 - val_accuracy: 0.7195
Epoch 472/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0526 -
accuracy: 0.6154 - val_loss: 0.8453 - val_accuracy: 0.7199
Epoch 473/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0321 -
accuracy: 0.6250 - val_loss: 0.8446 - val_accuracy: 0.7200
Epoch 474/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.0444 -
accuracy: 0.6204 - val_loss: 0.8443 - val_accuracy: 0.7203
Epoch 475/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0434 -
```

```
accuracy: 0.6252 - val_loss: 0.8442 - val_accuracy: 0.7206
Epoch 476/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0275 -
accuracy: 0.6248 - val_loss: 0.8433 - val_accuracy: 0.7208
Epoch 477/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0309 -
accuracy: 0.6214 - val_loss: 0.8421 - val_accuracy: 0.7211
Epoch 478/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0342 -
accuracy: 0.6205 - val_loss: 0.8415 - val_accuracy: 0.7217
Epoch 479/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0199 -
accuracy: 0.6300 - val_loss: 0.8409 - val_accuracy: 0.7226
Epoch 480/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.0335 -
accuracy: 0.6242 - val_loss: 0.8400 - val_accuracy: 0.7226
Epoch 481/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.0125 -
accuracy: 0.6304 - val_loss: 0.8394 - val_accuracy: 0.7222
Epoch 482/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0189 -
accuracy: 0.6269 - val_loss: 0.8380 - val_accuracy: 0.7226
Epoch 483/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0298 -
accuracy: 0.6226 - val_loss: 0.8377 - val_accuracy: 0.7229
Epoch 484/1000
9727/9727 [=====] - 1s 59us/sample - loss: 1.0288 -
accuracy: 0.6296 - val_loss: 0.8367 - val_accuracy: 0.7226
Epoch 485/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0244 -
accuracy: 0.6274 - val_loss: 0.8362 - val_accuracy: 0.7233
Epoch 486/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0232 -
accuracy: 0.6301 - val_loss: 0.8352 - val_accuracy: 0.7236
Epoch 487/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.0139 -
accuracy: 0.6327 - val_loss: 0.8348 - val_accuracy: 0.7234
Epoch 488/1000
9727/9727 [=====] - 1s 60us/sample - loss: 1.0181 -
accuracy: 0.6303 - val_loss: 0.8342 - val_accuracy: 0.7237
Epoch 489/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.0165 -
accuracy: 0.6300 - val_loss: 0.8331 - val_accuracy: 0.7254
Epoch 490/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0190 -
accuracy: 0.6291 - val_loss: 0.8322 - val_accuracy: 0.7246
Epoch 491/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0178 -
accuracy: 0.6346 - val_loss: 0.8312 - val_accuracy: 0.7240
Epoch 492/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0221 -
accuracy: 0.6302 - val_loss: 0.8302 - val_accuracy: 0.7240
Epoch 493/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.0157 -
accuracy: 0.6316 - val_loss: 0.8296 - val_accuracy: 0.7251
Epoch 494/1000
9727/9727 [=====] - 1s 56us/sample - loss: 1.0084 -
```

```
accuracy: 0.6360 - val_loss: 0.8288 - val_accuracy: 0.7254
Epoch 495/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0206 -
accuracy: 0.6338 - val_loss: 0.8279 - val_accuracy: 0.7256
Epoch 496/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0080 -
accuracy: 0.6318 - val_loss: 0.8273 - val_accuracy: 0.7254
Epoch 497/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0076 -
accuracy: 0.6308 - val_loss: 0.8262 - val_accuracy: 0.7256
Epoch 498/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0101 -
accuracy: 0.6375 - val_loss: 0.8261 - val_accuracy: 0.7263
Epoch 499/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0029 -
accuracy: 0.6381 - val_loss: 0.8254 - val_accuracy: 0.7271
Epoch 500/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0114 -
accuracy: 0.6381 - val_loss: 0.8243 - val_accuracy: 0.7270
Epoch 501/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0060 -
accuracy: 0.6322 - val_loss: 0.8239 - val_accuracy: 0.7279
Epoch 502/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0066 -
accuracy: 0.6377 - val_loss: 0.8233 - val_accuracy: 0.7263
Epoch 503/1000
9727/9727 [=====] - 1s 57us/sample - loss: 1.0145 -
accuracy: 0.6363 - val_loss: 0.8227 - val_accuracy: 0.7271
Epoch 504/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0066 -
accuracy: 0.6391 - val_loss: 0.8221 - val_accuracy: 0.7274
Epoch 505/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0126 -
accuracy: 0.6316 - val_loss: 0.8215 - val_accuracy: 0.7282
Epoch 506/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0064 -
accuracy: 0.6341 - val_loss: 0.8211 - val_accuracy: 0.7288
Epoch 507/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0022 -
accuracy: 0.6383 - val_loss: 0.8206 - val_accuracy: 0.7288
Epoch 508/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9991 -
accuracy: 0.6360 - val_loss: 0.8201 - val_accuracy: 0.7285
Epoch 509/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9999 -
accuracy: 0.6400 - val_loss: 0.8193 - val_accuracy: 0.7280
Epoch 510/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0021 -
accuracy: 0.6372 - val_loss: 0.8187 - val_accuracy: 0.7282
Epoch 511/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0075 -
accuracy: 0.6359 - val_loss: 0.8181 - val_accuracy: 0.7283
Epoch 512/1000
9727/9727 [=====] - 1s 54us/sample - loss: 1.0110 -
accuracy: 0.6341 - val_loss: 0.8186 - val_accuracy: 0.7296
Epoch 513/1000
9727/9727 [=====] - 1s 58us/sample - loss: 1.0018 -
```

```
accuracy: 0.6333 - val_loss: 0.8177 - val_accuracy: 0.7294
Epoch 514/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9905 -
accuracy: 0.6379 - val_loss: 0.8168 - val_accuracy: 0.7297
Epoch 515/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9981 -
accuracy: 0.6363 - val_loss: 0.8161 - val_accuracy: 0.7294
Epoch 516/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9935 -
accuracy: 0.6389 - val_loss: 0.8153 - val_accuracy: 0.7300
Epoch 517/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9983 -
accuracy: 0.6403 - val_loss: 0.8144 - val_accuracy: 0.7302
Epoch 518/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9985 -
accuracy: 0.6388 - val_loss: 0.8138 - val_accuracy: 0.7303
Epoch 519/1000
9727/9727 [=====] - 1s 55us/sample - loss: 1.0001 -
accuracy: 0.6364 - val_loss: 0.8135 - val_accuracy: 0.7303
Epoch 520/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9891 -
accuracy: 0.6431 - val_loss: 0.8129 - val_accuracy: 0.7313
Epoch 521/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9980 -
accuracy: 0.6407 - val_loss: 0.8124 - val_accuracy: 0.7305
Epoch 522/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9887 -
accuracy: 0.6416 - val_loss: 0.8112 - val_accuracy: 0.7316
Epoch 523/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.9968 -
accuracy: 0.6448 - val_loss: 0.8104 - val_accuracy: 0.7311
Epoch 524/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.9866 -
accuracy: 0.6442 - val_loss: 0.8096 - val_accuracy: 0.7310
Epoch 525/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9888 -
accuracy: 0.6411 - val_loss: 0.8091 - val_accuracy: 0.7313
Epoch 526/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9933 -
accuracy: 0.6329 - val_loss: 0.8089 - val_accuracy: 0.7311
Epoch 527/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9893 -
accuracy: 0.6426 - val_loss: 0.8085 - val_accuracy: 0.7319
Epoch 528/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.9821 -
accuracy: 0.6410 - val_loss: 0.8081 - val_accuracy: 0.7314
Epoch 529/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9889 -
accuracy: 0.6423 - val_loss: 0.8072 - val_accuracy: 0.7316
Epoch 530/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9922 -
accuracy: 0.6375 - val_loss: 0.8066 - val_accuracy: 0.7308
Epoch 531/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.9883 -
accuracy: 0.6448 - val_loss: 0.8061 - val_accuracy: 0.7310
Epoch 532/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9880 -
```

```
accuracy: 0.6408 - val_loss: 0.8052 - val_accuracy: 0.7316
Epoch 533/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9830 -
accuracy: 0.6431 - val_loss: 0.8044 - val_accuracy: 0.7314
Epoch 534/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9729 -
accuracy: 0.6459 - val_loss: 0.8035 - val_accuracy: 0.7323
Epoch 535/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9811 -
accuracy: 0.6459 - val_loss: 0.8027 - val_accuracy: 0.7320
Epoch 536/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9873 -
accuracy: 0.6475 - val_loss: 0.8022 - val_accuracy: 0.7325
Epoch 537/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9853 -
accuracy: 0.6426 - val_loss: 0.8021 - val_accuracy: 0.7331
Epoch 538/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9755 -
accuracy: 0.6521 - val_loss: 0.8011 - val_accuracy: 0.7337
Epoch 539/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9725 -
accuracy: 0.6543 - val_loss: 0.8002 - val_accuracy: 0.7334
Epoch 540/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9674 -
accuracy: 0.6500 - val_loss: 0.7992 - val_accuracy: 0.7328
Epoch 541/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9715 -
accuracy: 0.6548 - val_loss: 0.7982 - val_accuracy: 0.7339
Epoch 542/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9782 -
accuracy: 0.6448 - val_loss: 0.7981 - val_accuracy: 0.7345
Epoch 543/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9748 -
accuracy: 0.6483 - val_loss: 0.7975 - val_accuracy: 0.7344
Epoch 544/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9662 -
accuracy: 0.6537 - val_loss: 0.7966 - val_accuracy: 0.7342
Epoch 545/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.9819 -
accuracy: 0.6487 - val_loss: 0.7968 - val_accuracy: 0.7350
Epoch 546/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9695 -
accuracy: 0.6529 - val_loss: 0.7960 - val_accuracy: 0.7350
Epoch 547/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9681 -
accuracy: 0.6492 - val_loss: 0.7949 - val_accuracy: 0.7354
Epoch 548/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9733 -
accuracy: 0.6457 - val_loss: 0.7942 - val_accuracy: 0.7357
Epoch 549/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9594 -
accuracy: 0.6542 - val_loss: 0.7925 - val_accuracy: 0.7359
Epoch 550/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9648 -
accuracy: 0.6593 - val_loss: 0.7918 - val_accuracy: 0.7362
Epoch 551/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9723 -
```

```
accuracy: 0.6458 - val_loss: 0.7915 - val_accuracy: 0.7365
Epoch 552/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.9626 -
accuracy: 0.6551 - val_loss: 0.7903 - val_accuracy: 0.7360
Epoch 553/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9705 -
accuracy: 0.6544 - val_loss: 0.7895 - val_accuracy: 0.7373
Epoch 554/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.9645 -
accuracy: 0.6515 - val_loss: 0.7889 - val_accuracy: 0.7376
Epoch 555/1000
9727/9727 [=====] - 1s 62us/sample - loss: 0.9605 -
accuracy: 0.6521 - val_loss: 0.7886 - val_accuracy: 0.7365
Epoch 556/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9586 -
accuracy: 0.6528 - val_loss: 0.7881 - val_accuracy: 0.7365
Epoch 557/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9622 -
accuracy: 0.6522 - val_loss: 0.7872 - val_accuracy: 0.7370
Epoch 558/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9606 -
accuracy: 0.6558 - val_loss: 0.7866 - val_accuracy: 0.7382
Epoch 559/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9629 -
accuracy: 0.6572 - val_loss: 0.7862 - val_accuracy: 0.7382
Epoch 560/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.9670 -
accuracy: 0.6539 - val_loss: 0.7859 - val_accuracy: 0.7391
Epoch 561/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9734 -
accuracy: 0.6499 - val_loss: 0.7853 - val_accuracy: 0.7394
Epoch 562/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.9716 -
accuracy: 0.6517 - val_loss: 0.7847 - val_accuracy: 0.7393
Epoch 563/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9630 -
accuracy: 0.6488 - val_loss: 0.7848 - val_accuracy: 0.7402
Epoch 564/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9579 -
accuracy: 0.6521 - val_loss: 0.7841 - val_accuracy: 0.7399
Epoch 565/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9531 -
accuracy: 0.6565 - val_loss: 0.7836 - val_accuracy: 0.7394
Epoch 566/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9443 -
accuracy: 0.6615 - val_loss: 0.7828 - val_accuracy: 0.7399
Epoch 567/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9540 -
accuracy: 0.6567 - val_loss: 0.7819 - val_accuracy: 0.7399
Epoch 568/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9560 -
accuracy: 0.6554 - val_loss: 0.7815 - val_accuracy: 0.7401
Epoch 569/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9457 -
accuracy: 0.6640 - val_loss: 0.7799 - val_accuracy: 0.7405
Epoch 570/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9628 -
```



```
accuracy: 0.6558 - val_loss: 0.7797 - val_accuracy: 0.7399
Epoch 571/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9464 -
accuracy: 0.6598 - val_loss: 0.7794 - val_accuracy: 0.7405
Epoch 572/1000
9727/9727 [=====] - 1s 61us/sample - loss: 0.9552 -
accuracy: 0.6624 - val_loss: 0.7792 - val_accuracy: 0.7414
Epoch 573/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.9589 -
accuracy: 0.6540 - val_loss: 0.7786 - val_accuracy: 0.7405
Epoch 574/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9569 -
accuracy: 0.6579 - val_loss: 0.7778 - val_accuracy: 0.7411
Epoch 575/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.9449 -
accuracy: 0.6628 - val_loss: 0.7768 - val_accuracy: 0.7411
Epoch 576/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9457 -
accuracy: 0.6577 - val_loss: 0.7760 - val_accuracy: 0.7414
Epoch 577/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9486 -
accuracy: 0.6545 - val_loss: 0.7762 - val_accuracy: 0.7411
Epoch 578/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9377 -
accuracy: 0.6639 - val_loss: 0.7752 - val_accuracy: 0.7411
Epoch 579/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9609 -
accuracy: 0.6506 - val_loss: 0.7750 - val_accuracy: 0.7424
Epoch 580/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9494 -
accuracy: 0.6643 - val_loss: 0.7744 - val_accuracy: 0.7424
Epoch 581/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.9468 -
accuracy: 0.6537 - val_loss: 0.7734 - val_accuracy: 0.7422
Epoch 582/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9453 -
accuracy: 0.6607 - val_loss: 0.7728 - val_accuracy: 0.7424
Epoch 583/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9573 -
accuracy: 0.6606 - val_loss: 0.7729 - val_accuracy: 0.7425
Epoch 584/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9426 -
accuracy: 0.6644 - val_loss: 0.7723 - val_accuracy: 0.7425
Epoch 585/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9446 -
accuracy: 0.6627 - val_loss: 0.7714 - val_accuracy: 0.7424
Epoch 586/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9410 -
accuracy: 0.6609 - val_loss: 0.7706 - val_accuracy: 0.7434
Epoch 587/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9454 -
accuracy: 0.6624 - val_loss: 0.7702 - val_accuracy: 0.7428
Epoch 588/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9480 -
accuracy: 0.6608 - val_loss: 0.7700 - val_accuracy: 0.7430
Epoch 589/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9401 -
```

```
accuracy: 0.6610 - val_loss: 0.7691 - val_accuracy: 0.7441
Epoch 590/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.9453 -
accuracy: 0.6618 - val_loss: 0.7684 - val_accuracy: 0.7439
Epoch 591/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9331 -
accuracy: 0.6623 - val_loss: 0.7682 - val_accuracy: 0.7455
Epoch 592/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9413 -
accuracy: 0.6608 - val_loss: 0.7676 - val_accuracy: 0.7451
Epoch 593/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9410 -
accuracy: 0.6629 - val_loss: 0.7669 - val_accuracy: 0.7445
Epoch 594/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9417 -
accuracy: 0.6624 - val_loss: 0.7664 - val_accuracy: 0.7450
Epoch 595/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9348 -
accuracy: 0.6654 - val_loss: 0.7656 - val_accuracy: 0.7455
Epoch 596/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9250 -
accuracy: 0.6636 - val_loss: 0.7642 - val_accuracy: 0.7453
Epoch 597/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9393 -
accuracy: 0.6643 - val_loss: 0.7640 - val_accuracy: 0.7456
Epoch 598/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9465 -
accuracy: 0.6621 - val_loss: 0.7636 - val_accuracy: 0.7455
Epoch 599/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9441 -
accuracy: 0.6629 - val_loss: 0.7636 - val_accuracy: 0.7455
Epoch 600/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9436 -
accuracy: 0.6620 - val_loss: 0.7631 - val_accuracy: 0.7450
Epoch 601/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9337 -
accuracy: 0.6670 - val_loss: 0.7625 - val_accuracy: 0.7451
Epoch 602/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9303 -
accuracy: 0.6690 - val_loss: 0.7619 - val_accuracy: 0.7453
Epoch 603/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9268 -
accuracy: 0.6709 - val_loss: 0.7616 - val_accuracy: 0.7456
Epoch 604/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9391 -
accuracy: 0.6639 - val_loss: 0.7612 - val_accuracy: 0.7471
Epoch 605/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9259 -
accuracy: 0.6666 - val_loss: 0.7603 - val_accuracy: 0.7473
Epoch 606/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9317 -
accuracy: 0.6679 - val_loss: 0.7597 - val_accuracy: 0.7479
Epoch 607/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9300 -
accuracy: 0.6678 - val_loss: 0.7593 - val_accuracy: 0.7476
Epoch 608/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9385 -
```

```
accuracy: 0.6699 - val_loss: 0.7587 - val_accuracy: 0.7479
Epoch 609/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9385 -
accuracy: 0.6663 - val_loss: 0.7581 - val_accuracy: 0.7478
Epoch 610/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9256 -
accuracy: 0.6682 - val_loss: 0.7571 - val_accuracy: 0.7485
Epoch 611/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9230 -
accuracy: 0.6730 - val_loss: 0.7563 - val_accuracy: 0.7484
Epoch 612/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9180 -
accuracy: 0.6705 - val_loss: 0.7556 - val_accuracy: 0.7492
Epoch 613/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9189 -
accuracy: 0.6742 - val_loss: 0.7548 - val_accuracy: 0.7488
Epoch 614/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9263 -
accuracy: 0.6656 - val_loss: 0.7543 - val_accuracy: 0.7487
Epoch 615/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9110 -
accuracy: 0.6734 - val_loss: 0.7542 - val_accuracy: 0.7479
Epoch 616/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9280 -
accuracy: 0.6686 - val_loss: 0.7540 - val_accuracy: 0.7484
Epoch 617/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9245 -
accuracy: 0.6691 - val_loss: 0.7535 - val_accuracy: 0.7488
Epoch 618/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9176 -
accuracy: 0.6710 - val_loss: 0.7524 - val_accuracy: 0.7484
Epoch 619/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9222 -
accuracy: 0.6730 - val_loss: 0.7519 - val_accuracy: 0.7496
Epoch 620/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9177 -
accuracy: 0.6717 - val_loss: 0.7512 - val_accuracy: 0.7496
Epoch 621/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9084 -
accuracy: 0.6765 - val_loss: 0.7504 - val_accuracy: 0.7495
Epoch 622/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9125 -
accuracy: 0.6714 - val_loss: 0.7500 - val_accuracy: 0.7496
Epoch 623/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9242 -
accuracy: 0.6677 - val_loss: 0.7492 - val_accuracy: 0.7508
Epoch 624/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9177 -
accuracy: 0.6694 - val_loss: 0.7490 - val_accuracy: 0.7510
Epoch 625/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9160 -
accuracy: 0.6732 - val_loss: 0.7486 - val_accuracy: 0.7505
Epoch 626/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9037 -
accuracy: 0.6744 - val_loss: 0.7479 - val_accuracy: 0.7527
Epoch 627/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9157 -
```

```
accuracy: 0.6717 - val_loss: 0.7475 - val_accuracy: 0.7516
Epoch 628/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9127 -
accuracy: 0.6706 - val_loss: 0.7474 - val_accuracy: 0.7507
Epoch 629/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9179 -
accuracy: 0.6793 - val_loss: 0.7469 - val_accuracy: 0.7510
Epoch 630/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.9164 -
accuracy: 0.6750 - val_loss: 0.7462 - val_accuracy: 0.7513
Epoch 631/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9017 -
accuracy: 0.6750 - val_loss: 0.7455 - val_accuracy: 0.7527
Epoch 632/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8984 -
accuracy: 0.6789 - val_loss: 0.7448 - val_accuracy: 0.7535
Epoch 633/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9135 -
accuracy: 0.6772 - val_loss: 0.7443 - val_accuracy: 0.7535
Epoch 634/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9017 -
accuracy: 0.6820 - val_loss: 0.7431 - val_accuracy: 0.7536
Epoch 635/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9122 -
accuracy: 0.6784 - val_loss: 0.7426 - val_accuracy: 0.7539
Epoch 636/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9108 -
accuracy: 0.6712 - val_loss: 0.7419 - val_accuracy: 0.7542
Epoch 637/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9079 -
accuracy: 0.6791 - val_loss: 0.7413 - val_accuracy: 0.7544
Epoch 638/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9124 -
accuracy: 0.6746 - val_loss: 0.7417 - val_accuracy: 0.7542
Epoch 639/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9066 -
accuracy: 0.6754 - val_loss: 0.7413 - val_accuracy: 0.7552
Epoch 640/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.9135 -
accuracy: 0.6761 - val_loss: 0.7409 - val_accuracy: 0.7542
Epoch 641/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.9186 -
accuracy: 0.6691 - val_loss: 0.7407 - val_accuracy: 0.7541
Epoch 642/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9045 -
accuracy: 0.6730 - val_loss: 0.7397 - val_accuracy: 0.7547
Epoch 643/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9061 -
accuracy: 0.6739 - val_loss: 0.7392 - val_accuracy: 0.7549
Epoch 644/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9091 -
accuracy: 0.6754 - val_loss: 0.7381 - val_accuracy: 0.7561
Epoch 645/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8949 -
accuracy: 0.6816 - val_loss: 0.7376 - val_accuracy: 0.7555
Epoch 646/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8990 -
```

```
accuracy: 0.6801 - val_loss: 0.7369 - val_accuracy: 0.7556
Epoch 647/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9105 -
accuracy: 0.6711 - val_loss: 0.7364 - val_accuracy: 0.7561
Epoch 648/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8978 -
accuracy: 0.6775 - val_loss: 0.7353 - val_accuracy: 0.7567
Epoch 649/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8949 -
accuracy: 0.6811 - val_loss: 0.7345 - val_accuracy: 0.7559
Epoch 650/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.9014 -
accuracy: 0.6801 - val_loss: 0.7343 - val_accuracy: 0.7569
Epoch 651/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.8930 -
accuracy: 0.6851 - val_loss: 0.7337 - val_accuracy: 0.7570
Epoch 652/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8931 -
accuracy: 0.6822 - val_loss: 0.7329 - val_accuracy: 0.7570
Epoch 653/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.9047 -
accuracy: 0.6776 - val_loss: 0.7328 - val_accuracy: 0.7576
Epoch 654/1000
9727/9727 [=====] - 1s 59us/sample - loss: 0.8865 -
accuracy: 0.6826 - val_loss: 0.7322 - val_accuracy: 0.7575
Epoch 655/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.9041 -
accuracy: 0.6764 - val_loss: 0.7319 - val_accuracy: 0.7569
Epoch 656/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8996 -
accuracy: 0.6801 - val_loss: 0.7313 - val_accuracy: 0.7561
Epoch 657/1000
9727/9727 [=====] - 1s 52us/sample - loss: 0.8977 -
accuracy: 0.6791 - val_loss: 0.7313 - val_accuracy: 0.7572
Epoch 658/1000
9727/9727 [=====] - 1s 59us/sample - loss: 0.9003 -
accuracy: 0.6794 - val_loss: 0.7308 - val_accuracy: 0.7567
Epoch 659/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8831 -
accuracy: 0.6853 - val_loss: 0.7307 - val_accuracy: 0.7572
Epoch 660/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.9021 -
accuracy: 0.6780 - val_loss: 0.7304 - val_accuracy: 0.7584
Epoch 661/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8984 -
accuracy: 0.6849 - val_loss: 0.7303 - val_accuracy: 0.7603
Epoch 662/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8966 -
accuracy: 0.6806 - val_loss: 0.7297 - val_accuracy: 0.7596
Epoch 663/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8855 -
accuracy: 0.6897 - val_loss: 0.7288 - val_accuracy: 0.7596
Epoch 664/1000
9727/9727 [=====] - 1s 61us/sample - loss: 0.8866 -
accuracy: 0.6872 - val_loss: 0.7282 - val_accuracy: 0.7589
Epoch 665/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8912 -
```

```
accuracy: 0.6854 - val_loss: 0.7275 - val_accuracy: 0.7586
Epoch 666/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8970 -
accuracy: 0.6782 - val_loss: 0.7271 - val_accuracy: 0.7589
Epoch 667/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8891 -
accuracy: 0.6801 - val_loss: 0.7268 - val_accuracy: 0.7589
Epoch 668/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8859 -
accuracy: 0.6826 - val_loss: 0.7264 - val_accuracy: 0.7599
Epoch 669/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.8966 -
accuracy: 0.6799 - val_loss: 0.7259 - val_accuracy: 0.7604
Epoch 670/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8930 -
accuracy: 0.6838 - val_loss: 0.7261 - val_accuracy: 0.7609
Epoch 671/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8958 -
accuracy: 0.6817 - val_loss: 0.7257 - val_accuracy: 0.7604
Epoch 672/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8989 -
accuracy: 0.6828 - val_loss: 0.7254 - val_accuracy: 0.7609
Epoch 673/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8886 -
accuracy: 0.6850 - val_loss: 0.7246 - val_accuracy: 0.7613
Epoch 674/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8831 -
accuracy: 0.6855 - val_loss: 0.7236 - val_accuracy: 0.7606
Epoch 675/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8739 -
accuracy: 0.6896 - val_loss: 0.7231 - val_accuracy: 0.7604
Epoch 676/1000
9727/9727 [=====] - 1s 52us/sample - loss: 0.8907 -
accuracy: 0.6805 - val_loss: 0.7228 - val_accuracy: 0.7613
Epoch 677/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8837 -
accuracy: 0.6887 - val_loss: 0.7225 - val_accuracy: 0.7613
Epoch 678/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8854 -
accuracy: 0.6848 - val_loss: 0.7218 - val_accuracy: 0.7615
Epoch 679/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8893 -
accuracy: 0.6826 - val_loss: 0.7217 - val_accuracy: 0.7604
Epoch 680/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.8790 -
accuracy: 0.6876 - val_loss: 0.7214 - val_accuracy: 0.7621
Epoch 681/1000
9727/9727 [=====] - 1s 52us/sample - loss: 0.8724 -
accuracy: 0.6989 - val_loss: 0.7211 - val_accuracy: 0.7623
Epoch 682/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8849 -
accuracy: 0.6864 - val_loss: 0.7209 - val_accuracy: 0.7609
Epoch 683/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8744 -
accuracy: 0.6901 - val_loss: 0.7197 - val_accuracy: 0.7623
Epoch 684/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8781 -
```

```
accuracy: 0.6929 - val_loss: 0.7193 - val_accuracy: 0.7626
Epoch 685/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8636 -
accuracy: 0.6903 - val_loss: 0.7186 - val_accuracy: 0.7627
Epoch 686/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8727 -
accuracy: 0.6902 - val_loss: 0.7181 - val_accuracy: 0.7633
Epoch 687/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8786 -
accuracy: 0.6923 - val_loss: 0.7173 - val_accuracy: 0.7640
Epoch 688/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8755 -
accuracy: 0.6877 - val_loss: 0.7170 - val_accuracy: 0.7635
Epoch 689/1000
9727/9727 [=====] - 1s 59us/sample - loss: 0.8724 -
accuracy: 0.6908 - val_loss: 0.7165 - val_accuracy: 0.7641
Epoch 690/1000
9727/9727 [=====] - 1s 64us/sample - loss: 0.8704 -
accuracy: 0.6916 - val_loss: 0.7160 - val_accuracy: 0.7640
Epoch 691/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8808 -
accuracy: 0.6862 - val_loss: 0.7159 - val_accuracy: 0.7640
Epoch 692/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8807 -
accuracy: 0.6866 - val_loss: 0.7154 - val_accuracy: 0.7647
Epoch 693/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8793 -
accuracy: 0.6928 - val_loss: 0.7149 - val_accuracy: 0.7647
Epoch 694/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8731 -
accuracy: 0.6897 - val_loss: 0.7143 - val_accuracy: 0.7653
Epoch 695/1000
9727/9727 [=====] - 1s 52us/sample - loss: 0.8790 -
accuracy: 0.6879 - val_loss: 0.7138 - val_accuracy: 0.7647
Epoch 696/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8770 -
accuracy: 0.6936 - val_loss: 0.7132 - val_accuracy: 0.7656
Epoch 697/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8656 -
accuracy: 0.6943 - val_loss: 0.7131 - val_accuracy: 0.7649
Epoch 698/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8665 -
accuracy: 0.6893 - val_loss: 0.7121 - val_accuracy: 0.7647
Epoch 699/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8759 -
accuracy: 0.6883 - val_loss: 0.7112 - val_accuracy: 0.7644
Epoch 700/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.8532 -
accuracy: 0.7034 - val_loss: 0.7104 - val_accuracy: 0.7646
Epoch 701/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8740 -
accuracy: 0.6930 - val_loss: 0.7098 - val_accuracy: 0.7656
Epoch 702/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8788 -
accuracy: 0.6861 - val_loss: 0.7095 - val_accuracy: 0.7658
Epoch 703/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8648 -
```

```
accuracy: 0.6973 - val_loss: 0.7086 - val_accuracy: 0.7661
Epoch 704/1000
9727/9727 [=====] - 1s 52us/sample - loss: 0.8494 -
accuracy: 0.7014 - val_loss: 0.7076 - val_accuracy: 0.7664
Epoch 705/1000
9727/9727 [=====] - 1s 52us/sample - loss: 0.8560 -
accuracy: 0.6977 - val_loss: 0.7072 - val_accuracy: 0.7663
Epoch 706/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8657 -
accuracy: 0.6963 - val_loss: 0.7070 - val_accuracy: 0.7663
Epoch 707/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8682 -
accuracy: 0.6931 - val_loss: 0.7064 - val_accuracy: 0.7667
Epoch 708/1000
9727/9727 [=====] - 1s 59us/sample - loss: 0.8622 -
accuracy: 0.6925 - val_loss: 0.7061 - val_accuracy: 0.7670
Epoch 709/1000
9727/9727 [=====] - 1s 65us/sample - loss: 0.8603 -
accuracy: 0.6982 - val_loss: 0.7051 - val_accuracy: 0.7670
Epoch 710/1000
9727/9727 [=====] - 1s 86us/sample - loss: 0.8676 -
accuracy: 0.6917 - val_loss: 0.7051 - val_accuracy: 0.7669
Epoch 711/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8654 -
accuracy: 0.6946 - val_loss: 0.7049 - val_accuracy: 0.7678
Epoch 712/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8534 -
accuracy: 0.6982 - val_loss: 0.7040 - val_accuracy: 0.7678
Epoch 713/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8694 -
accuracy: 0.6922 - val_loss: 0.7037 - val_accuracy: 0.7686
Epoch 714/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8614 -
accuracy: 0.6959 - val_loss: 0.7033 - val_accuracy: 0.7678
Epoch 715/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.8571 -
accuracy: 0.6957 - val_loss: 0.7031 - val_accuracy: 0.7689
Epoch 716/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8641 -
accuracy: 0.6949 - val_loss: 0.7028 - val_accuracy: 0.7689
Epoch 717/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8627 -
accuracy: 0.6946 - val_loss: 0.7021 - val_accuracy: 0.7689
Epoch 718/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8650 -
accuracy: 0.6952 - val_loss: 0.7017 - val_accuracy: 0.7693
Epoch 719/1000
9727/9727 [=====] - 1s 59us/sample - loss: 0.8494 -
accuracy: 0.6994 - val_loss: 0.7010 - val_accuracy: 0.7695
Epoch 720/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8648 -
accuracy: 0.6954 - val_loss: 0.7011 - val_accuracy: 0.7697
Epoch 721/1000
9727/9727 [=====] - 1s 60us/sample - loss: 0.8717 -
accuracy: 0.6921 - val_loss: 0.7005 - val_accuracy: 0.7700
Epoch 722/1000
9727/9727 [=====] - 1s 69us/sample - loss: 0.8525 -
```



```
accuracy: 0.6987 - val_loss: 0.6997 - val_accuracy: 0.7701
Epoch 723/1000
9727/9727 [=====] - 1s 65us/sample - loss: 0.8589 -
accuracy: 0.6994 - val_loss: 0.6993 - val_accuracy: 0.7701
Epoch 724/1000
9727/9727 [=====] - 1s 60us/sample - loss: 0.8503 -
accuracy: 0.7010 - val_loss: 0.6989 - val_accuracy: 0.7693
Epoch 725/1000
9727/9727 [=====] - 1s 61us/sample - loss: 0.8469 -
accuracy: 0.6994 - val_loss: 0.6982 - val_accuracy: 0.7697
Epoch 726/1000
9727/9727 [=====] - 1s 66us/sample - loss: 0.8485 -
accuracy: 0.6971 - val_loss: 0.6974 - val_accuracy: 0.7712
Epoch 727/1000
9727/9727 [=====] - 1s 61us/sample - loss: 0.8535 -
accuracy: 0.6956 - val_loss: 0.6965 - val_accuracy: 0.7718
Epoch 728/1000
9727/9727 [=====] - 1s 61us/sample - loss: 0.8501 -
accuracy: 0.6990 - val_loss: 0.6956 - val_accuracy: 0.7724
Epoch 729/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8541 -
accuracy: 0.6983 - val_loss: 0.6954 - val_accuracy: 0.7720
Epoch 730/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8389 -
accuracy: 0.7090 - val_loss: 0.6950 - val_accuracy: 0.7721
Epoch 731/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8424 -
accuracy: 0.7007 - val_loss: 0.6944 - val_accuracy: 0.7718
Epoch 732/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8432 -
accuracy: 0.7064 - val_loss: 0.6940 - val_accuracy: 0.7715
Epoch 733/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8510 -
accuracy: 0.6987 - val_loss: 0.6937 - val_accuracy: 0.7721
Epoch 734/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8418 -
accuracy: 0.7039 - val_loss: 0.6932 - val_accuracy: 0.7720
Epoch 735/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8592 -
accuracy: 0.6953 - val_loss: 0.6931 - val_accuracy: 0.7727
Epoch 736/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8511 -
accuracy: 0.7050 - val_loss: 0.6924 - val_accuracy: 0.7737
Epoch 737/1000
9727/9727 [=====] - 1s 61us/sample - loss: 0.8466 -
accuracy: 0.7010 - val_loss: 0.6917 - val_accuracy: 0.7729
Epoch 738/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8406 -
accuracy: 0.7003 - val_loss: 0.6914 - val_accuracy: 0.7735
Epoch 739/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8433 -
accuracy: 0.7068 - val_loss: 0.6908 - val_accuracy: 0.7743
Epoch 740/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8524 -
accuracy: 0.6991 - val_loss: 0.6907 - val_accuracy: 0.7741
Epoch 741/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8448 -
```

```
accuracy: 0.7009 - val_loss: 0.6903 - val_accuracy: 0.7744
Epoch 742/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8347 -
accuracy: 0.7017 - val_loss: 0.6894 - val_accuracy: 0.7751
Epoch 743/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8394 -
accuracy: 0.7019 - val_loss: 0.6892 - val_accuracy: 0.7746
Epoch 744/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8356 -
accuracy: 0.7073 - val_loss: 0.6886 - val_accuracy: 0.7751
Epoch 745/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8460 -
accuracy: 0.7022 - val_loss: 0.6879 - val_accuracy: 0.7755
Epoch 746/1000
9727/9727 [=====] - 1s 62us/sample - loss: 0.8332 -
accuracy: 0.7033 - val_loss: 0.6874 - val_accuracy: 0.7761
Epoch 747/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8488 -
accuracy: 0.7025 - val_loss: 0.6875 - val_accuracy: 0.7755
Epoch 748/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8481 -
accuracy: 0.6951 - val_loss: 0.6873 - val_accuracy: 0.7754
Epoch 749/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8362 -
accuracy: 0.7058 - val_loss: 0.6868 - val_accuracy: 0.7757
Epoch 750/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8389 -
accuracy: 0.7042 - val_loss: 0.6869 - val_accuracy: 0.7754
Epoch 751/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.8279 -
accuracy: 0.7111 - val_loss: 0.6866 - val_accuracy: 0.7747
Epoch 752/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.8432 -
accuracy: 0.7067 - val_loss: 0.6862 - val_accuracy: 0.7749
Epoch 753/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8413 -
accuracy: 0.7042 - val_loss: 0.6861 - val_accuracy: 0.7751
Epoch 754/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.8372 -
accuracy: 0.7039 - val_loss: 0.6855 - val_accuracy: 0.7754
Epoch 755/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8315 -
accuracy: 0.7044 - val_loss: 0.6847 - val_accuracy: 0.7749
Epoch 756/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8324 -
accuracy: 0.7044 - val_loss: 0.6843 - val_accuracy: 0.7754
Epoch 757/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8406 -
accuracy: 0.7081 - val_loss: 0.6838 - val_accuracy: 0.7758
Epoch 758/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8274 -
accuracy: 0.7034 - val_loss: 0.6832 - val_accuracy: 0.7758
Epoch 759/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8430 -
accuracy: 0.7034 - val_loss: 0.6833 - val_accuracy: 0.7764
Epoch 760/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8326 -
```

```
accuracy: 0.7103 - val_loss: 0.6826 - val_accuracy: 0.7758
Epoch 761/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8241 -
accuracy: 0.7091 - val_loss: 0.6819 - val_accuracy: 0.7767
Epoch 762/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8233 -
accuracy: 0.7078 - val_loss: 0.6817 - val_accuracy: 0.7764
Epoch 763/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8319 -
accuracy: 0.7083 - val_loss: 0.6814 - val_accuracy: 0.7769
Epoch 764/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8392 -
accuracy: 0.7079 - val_loss: 0.6808 - val_accuracy: 0.7772
Epoch 765/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8277 -
accuracy: 0.7112 - val_loss: 0.6802 - val_accuracy: 0.7767
Epoch 766/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8301 -
accuracy: 0.7107 - val_loss: 0.6796 - val_accuracy: 0.7777
Epoch 767/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8282 -
accuracy: 0.7045 - val_loss: 0.6789 - val_accuracy: 0.7781
Epoch 768/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8354 -
accuracy: 0.7037 - val_loss: 0.6784 - val_accuracy: 0.7781
Epoch 769/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8327 -
accuracy: 0.7079 - val_loss: 0.6781 - val_accuracy: 0.7792
Epoch 770/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8310 -
accuracy: 0.7120 - val_loss: 0.6777 - val_accuracy: 0.7780
Epoch 771/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8272 -
accuracy: 0.7060 - val_loss: 0.6774 - val_accuracy: 0.7775
Epoch 772/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8324 -
accuracy: 0.7029 - val_loss: 0.6772 - val_accuracy: 0.7774
Epoch 773/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8349 -
accuracy: 0.7084 - val_loss: 0.6766 - val_accuracy: 0.7777
Epoch 774/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8310 -
accuracy: 0.7047 - val_loss: 0.6763 - val_accuracy: 0.7784
Epoch 775/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8127 -
accuracy: 0.7159 - val_loss: 0.6755 - val_accuracy: 0.7786
Epoch 776/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8285 -
accuracy: 0.7113 - val_loss: 0.6751 - val_accuracy: 0.7784
Epoch 777/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8229 -
accuracy: 0.7144 - val_loss: 0.6746 - val_accuracy: 0.7789
Epoch 778/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8206 -
accuracy: 0.7133 - val_loss: 0.6741 - val_accuracy: 0.7788
Epoch 779/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8294 -
```

```
accuracy: 0.7070 - val_loss: 0.6738 - val_accuracy: 0.7791
Epoch 780/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8168 -
accuracy: 0.7126 - val_loss: 0.6732 - val_accuracy: 0.7797
Epoch 781/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8110 -
accuracy: 0.7169 - val_loss: 0.6724 - val_accuracy: 0.7789
Epoch 782/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8212 -
accuracy: 0.7139 - val_loss: 0.6722 - val_accuracy: 0.7798
Epoch 783/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8172 -
accuracy: 0.7120 - val_loss: 0.6716 - val_accuracy: 0.7798
Epoch 784/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8281 -
accuracy: 0.7131 - val_loss: 0.6712 - val_accuracy: 0.7801
Epoch 785/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8137 -
accuracy: 0.7120 - val_loss: 0.6708 - val_accuracy: 0.7798
Epoch 786/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8182 -
accuracy: 0.7091 - val_loss: 0.6707 - val_accuracy: 0.7806
Epoch 787/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8274 -
accuracy: 0.7040 - val_loss: 0.6705 - val_accuracy: 0.7797
Epoch 788/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8206 -
accuracy: 0.7127 - val_loss: 0.6699 - val_accuracy: 0.7797
Epoch 789/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8125 -
accuracy: 0.7113 - val_loss: 0.6694 - val_accuracy: 0.7800
Epoch 790/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8146 -
accuracy: 0.7141 - val_loss: 0.6687 - val_accuracy: 0.7803
Epoch 791/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8116 -
accuracy: 0.7142 - val_loss: 0.6683 - val_accuracy: 0.7803
Epoch 792/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8131 -
accuracy: 0.7179 - val_loss: 0.6678 - val_accuracy: 0.7808
Epoch 793/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8252 -
accuracy: 0.7094 - val_loss: 0.6672 - val_accuracy: 0.7815
Epoch 794/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8137 -
accuracy: 0.7127 - val_loss: 0.6670 - val_accuracy: 0.7815
Epoch 795/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8185 -
accuracy: 0.7110 - val_loss: 0.6668 - val_accuracy: 0.7812
Epoch 796/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8126 -
accuracy: 0.7134 - val_loss: 0.6661 - val_accuracy: 0.7812
Epoch 797/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8175 -
accuracy: 0.7123 - val_loss: 0.6657 - val_accuracy: 0.7823
Epoch 798/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8046 -
```

```
accuracy: 0.7183 - val_loss: 0.6651 - val_accuracy: 0.7817
Epoch 799/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8184 -
accuracy: 0.7155 - val_loss: 0.6648 - val_accuracy: 0.7826
Epoch 800/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8117 -
accuracy: 0.7178 - val_loss: 0.6645 - val_accuracy: 0.7828
Epoch 801/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8090 -
accuracy: 0.7190 - val_loss: 0.6643 - val_accuracy: 0.7826
Epoch 802/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8059 -
accuracy: 0.7210 - val_loss: 0.6638 - val_accuracy: 0.7818
Epoch 803/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8026 -
accuracy: 0.7192 - val_loss: 0.6629 - val_accuracy: 0.7821
Epoch 804/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8110 -
accuracy: 0.7163 - val_loss: 0.6628 - val_accuracy: 0.7823
Epoch 805/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.8014 -
accuracy: 0.7186 - val_loss: 0.6625 - val_accuracy: 0.7829
Epoch 806/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8096 -
accuracy: 0.7160 - val_loss: 0.6620 - val_accuracy: 0.7828
Epoch 807/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8187 -
accuracy: 0.7136 - val_loss: 0.6622 - val_accuracy: 0.7831
Epoch 808/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7922 -
accuracy: 0.7252 - val_loss: 0.6619 - val_accuracy: 0.7828
Epoch 809/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8139 -
accuracy: 0.7104 - val_loss: 0.6611 - val_accuracy: 0.7837
Epoch 810/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7993 -
accuracy: 0.7183 - val_loss: 0.6606 - val_accuracy: 0.7837
Epoch 811/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.8139 -
accuracy: 0.7079 - val_loss: 0.6603 - val_accuracy: 0.7829
Epoch 812/1000
9727/9727 [=====] - 1s 59us/sample - loss: 0.8065 -
accuracy: 0.7132 - val_loss: 0.6600 - val_accuracy: 0.7832
Epoch 813/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.7963 -
accuracy: 0.7203 - val_loss: 0.6597 - val_accuracy: 0.7831
Epoch 814/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.8020 -
accuracy: 0.7160 - val_loss: 0.6592 - val_accuracy: 0.7834
Epoch 815/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.8072 -
accuracy: 0.7176 - val_loss: 0.6588 - val_accuracy: 0.7831
Epoch 816/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8064 -
accuracy: 0.7165 - val_loss: 0.6586 - val_accuracy: 0.7834
Epoch 817/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7916 -
```

```
accuracy: 0.7240 - val_loss: 0.6579 - val_accuracy: 0.7835
Epoch 818/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7923 -
accuracy: 0.7242 - val_loss: 0.6572 - val_accuracy: 0.7837
Epoch 819/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8022 -
accuracy: 0.7190 - val_loss: 0.6567 - val_accuracy: 0.7843
Epoch 820/1000
9727/9727 [=====] - 1s 60us/sample - loss: 0.7895 -
accuracy: 0.7254 - val_loss: 0.6567 - val_accuracy: 0.7842
Epoch 821/1000
9727/9727 [=====] - 1s 59us/sample - loss: 0.7981 -
accuracy: 0.7236 - val_loss: 0.6560 - val_accuracy: 0.7842
Epoch 822/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7976 -
accuracy: 0.7208 - val_loss: 0.6555 - val_accuracy: 0.7843
Epoch 823/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7984 -
accuracy: 0.7181 - val_loss: 0.6550 - val_accuracy: 0.7846
Epoch 824/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7997 -
accuracy: 0.7216 - val_loss: 0.6548 - val_accuracy: 0.7845
Epoch 825/1000
9727/9727 [=====] - 1s 59us/sample - loss: 0.7983 -
accuracy: 0.7175 - val_loss: 0.6542 - val_accuracy: 0.7846
Epoch 826/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.7963 -
accuracy: 0.7201 - val_loss: 0.6538 - val_accuracy: 0.7846
Epoch 827/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7967 -
accuracy: 0.7210 - val_loss: 0.6533 - val_accuracy: 0.7849
Epoch 828/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.8000 -
accuracy: 0.7231 - val_loss: 0.6533 - val_accuracy: 0.7849
Epoch 829/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7978 -
accuracy: 0.7186 - val_loss: 0.6528 - val_accuracy: 0.7854
Epoch 830/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.7986 -
accuracy: 0.7197 - val_loss: 0.6526 - val_accuracy: 0.7855
Epoch 831/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7942 -
accuracy: 0.7264 - val_loss: 0.6523 - val_accuracy: 0.7854
Epoch 832/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7953 -
accuracy: 0.7222 - val_loss: 0.6518 - val_accuracy: 0.7852
Epoch 833/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.8016 -
accuracy: 0.7207 - val_loss: 0.6514 - val_accuracy: 0.7854
Epoch 834/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7911 -
accuracy: 0.7206 - val_loss: 0.6510 - val_accuracy: 0.7855
Epoch 835/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7933 -
accuracy: 0.7236 - val_loss: 0.6504 - val_accuracy: 0.7855
Epoch 836/1000
9727/9727 [=====] - 1s 59us/sample - loss: 0.7914 -
```

```
accuracy: 0.7251 - val_loss: 0.6504 - val_accuracy: 0.7852
Epoch 837/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7924 -
accuracy: 0.7221 - val_loss: 0.6499 - val_accuracy: 0.7855
Epoch 838/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7909 -
accuracy: 0.7263 - val_loss: 0.6493 - val_accuracy: 0.7866
Epoch 839/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.7873 -
accuracy: 0.7244 - val_loss: 0.6486 - val_accuracy: 0.7862
Epoch 840/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7988 -
accuracy: 0.7212 - val_loss: 0.6482 - val_accuracy: 0.7865
Epoch 841/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7900 -
accuracy: 0.7253 - val_loss: 0.6480 - val_accuracy: 0.7858
Epoch 842/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7899 -
accuracy: 0.7240 - val_loss: 0.6474 - val_accuracy: 0.7869
Epoch 843/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.8008 -
accuracy: 0.7225 - val_loss: 0.6473 - val_accuracy: 0.7858
Epoch 844/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.7918 -
accuracy: 0.7209 - val_loss: 0.6469 - val_accuracy: 0.7879
Epoch 845/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7888 -
accuracy: 0.7255 - val_loss: 0.6465 - val_accuracy: 0.7869
Epoch 846/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7904 -
accuracy: 0.7203 - val_loss: 0.6466 - val_accuracy: 0.7877
Epoch 847/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7849 -
accuracy: 0.7247 - val_loss: 0.6461 - val_accuracy: 0.7874
Epoch 848/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7743 -
accuracy: 0.7295 - val_loss: 0.6454 - val_accuracy: 0.7874
Epoch 849/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7846 -
accuracy: 0.7239 - val_loss: 0.6451 - val_accuracy: 0.7868
Epoch 850/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7943 -
accuracy: 0.7183 - val_loss: 0.6447 - val_accuracy: 0.7879
Epoch 851/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7929 -
accuracy: 0.7227 - val_loss: 0.6445 - val_accuracy: 0.7880
Epoch 852/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7820 -
accuracy: 0.7267 - val_loss: 0.6438 - val_accuracy: 0.7880
Epoch 853/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7887 -
accuracy: 0.7224 - val_loss: 0.6434 - val_accuracy: 0.7889
Epoch 854/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7887 -
accuracy: 0.7238 - val_loss: 0.6431 - val_accuracy: 0.7891
Epoch 855/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7855 -
```

```
accuracy: 0.7252 - val_loss: 0.6425 - val_accuracy: 0.7886
Epoch 856/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7873 -
accuracy: 0.7236 - val_loss: 0.6422 - val_accuracy: 0.7885
Epoch 857/1000
9727/9727 [=====] - 1s 60us/sample - loss: 0.7861 -
accuracy: 0.7288 - val_loss: 0.6418 - val_accuracy: 0.7894
Epoch 858/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7954 -
accuracy: 0.7268 - val_loss: 0.6421 - val_accuracy: 0.7883
Epoch 859/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7845 -
accuracy: 0.7291 - val_loss: 0.6416 - val_accuracy: 0.7885
Epoch 860/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.7766 -
accuracy: 0.7302 - val_loss: 0.6411 - val_accuracy: 0.7879
Epoch 861/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7768 -
accuracy: 0.7255 - val_loss: 0.6406 - val_accuracy: 0.7885
Epoch 862/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7718 -
accuracy: 0.7270 - val_loss: 0.6403 - val_accuracy: 0.7888
Epoch 863/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.7833 -
accuracy: 0.7270 - val_loss: 0.6398 - val_accuracy: 0.7888
Epoch 864/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.7842 -
accuracy: 0.7298 - val_loss: 0.6391 - val_accuracy: 0.7894
Epoch 865/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7710 -
accuracy: 0.7288 - val_loss: 0.6386 - val_accuracy: 0.7892
Epoch 866/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7879 -
accuracy: 0.7261 - val_loss: 0.6384 - val_accuracy: 0.7894
Epoch 867/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7716 -
accuracy: 0.7326 - val_loss: 0.6379 - val_accuracy: 0.7894
Epoch 868/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7789 -
accuracy: 0.7255 - val_loss: 0.6378 - val_accuracy: 0.7891
Epoch 869/1000
9727/9727 [=====] - 1s 52us/sample - loss: 0.7749 -
accuracy: 0.7273 - val_loss: 0.6373 - val_accuracy: 0.7894
Epoch 870/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7821 -
accuracy: 0.7247 - val_loss: 0.6371 - val_accuracy: 0.7899
Epoch 871/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7664 -
accuracy: 0.7343 - val_loss: 0.6364 - val_accuracy: 0.7899
Epoch 872/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7686 -
accuracy: 0.7348 - val_loss: 0.6359 - val_accuracy: 0.7897
Epoch 873/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7766 -
accuracy: 0.7274 - val_loss: 0.6356 - val_accuracy: 0.7903
Epoch 874/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7734 -
```



```
accuracy: 0.7300 - val_loss: 0.6351 - val_accuracy: 0.7902
Epoch 875/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7560 -
accuracy: 0.7351 - val_loss: 0.6341 - val_accuracy: 0.7903
Epoch 876/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7782 -
accuracy: 0.7284 - val_loss: 0.6340 - val_accuracy: 0.7900
Epoch 877/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7621 -
accuracy: 0.7331 - val_loss: 0.6335 - val_accuracy: 0.7906
Epoch 878/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7728 -
accuracy: 0.7322 - val_loss: 0.6332 - val_accuracy: 0.7905
Epoch 879/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7685 -
accuracy: 0.7306 - val_loss: 0.6326 - val_accuracy: 0.7914
Epoch 880/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7696 -
accuracy: 0.7316 - val_loss: 0.6324 - val_accuracy: 0.7917
Epoch 881/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7644 -
accuracy: 0.7313 - val_loss: 0.6322 - val_accuracy: 0.7914
Epoch 882/1000
9727/9727 [=====] - 1s 52us/sample - loss: 0.7756 -
accuracy: 0.7341 - val_loss: 0.6323 - val_accuracy: 0.7912
Epoch 883/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7714 -
accuracy: 0.7272 - val_loss: 0.6316 - val_accuracy: 0.7906
Epoch 884/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7761 -
accuracy: 0.7325 - val_loss: 0.6313 - val_accuracy: 0.7916
Epoch 885/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.7768 -
accuracy: 0.7323 - val_loss: 0.6312 - val_accuracy: 0.7911
Epoch 886/1000
9727/9727 [=====] - 1s 52us/sample - loss: 0.7659 -
accuracy: 0.7320 - val_loss: 0.6306 - val_accuracy: 0.7919
Epoch 887/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7537 -
accuracy: 0.7398 - val_loss: 0.6301 - val_accuracy: 0.7922
Epoch 888/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7668 -
accuracy: 0.7349 - val_loss: 0.6295 - val_accuracy: 0.7923
Epoch 889/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7718 -
accuracy: 0.7295 - val_loss: 0.6296 - val_accuracy: 0.7922
Epoch 890/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7657 -
accuracy: 0.7307 - val_loss: 0.6292 - val_accuracy: 0.7920
Epoch 891/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7715 -
accuracy: 0.7302 - val_loss: 0.6288 - val_accuracy: 0.7931
Epoch 892/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7593 -
accuracy: 0.7372 - val_loss: 0.6284 - val_accuracy: 0.7922
Epoch 893/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7595 -
```

```
accuracy: 0.7327 - val_loss: 0.6279 - val_accuracy: 0.7929
Epoch 894/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7559 -
accuracy: 0.7372 - val_loss: 0.6273 - val_accuracy: 0.7923
Epoch 895/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7625 -
accuracy: 0.7356 - val_loss: 0.6271 - val_accuracy: 0.7925
Epoch 896/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7659 -
accuracy: 0.7369 - val_loss: 0.6268 - val_accuracy: 0.7931
Epoch 897/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7736 -
accuracy: 0.7331 - val_loss: 0.6269 - val_accuracy: 0.7928
Epoch 898/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7755 -
accuracy: 0.7267 - val_loss: 0.6267 - val_accuracy: 0.7931
Epoch 899/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7774 -
accuracy: 0.7302 - val_loss: 0.6263 - val_accuracy: 0.7932
Epoch 900/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7662 -
accuracy: 0.7309 - val_loss: 0.6261 - val_accuracy: 0.7932
Epoch 901/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7579 -
accuracy: 0.7316 - val_loss: 0.6257 - val_accuracy: 0.7940
Epoch 902/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7615 -
accuracy: 0.7355 - val_loss: 0.6252 - val_accuracy: 0.7942
Epoch 903/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.7596 -
accuracy: 0.7337 - val_loss: 0.6248 - val_accuracy: 0.7934
Epoch 904/1000
9727/9727 [=====] - 1s 62us/sample - loss: 0.7576 -
accuracy: 0.7388 - val_loss: 0.6243 - val_accuracy: 0.7936
Epoch 905/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7535 -
accuracy: 0.7345 - val_loss: 0.6241 - val_accuracy: 0.7931
Epoch 906/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7598 -
accuracy: 0.7332 - val_loss: 0.6239 - val_accuracy: 0.7934
Epoch 907/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7605 -
accuracy: 0.7357 - val_loss: 0.6237 - val_accuracy: 0.7939
Epoch 908/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7625 -
accuracy: 0.7397 - val_loss: 0.6235 - val_accuracy: 0.7934
Epoch 909/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7620 -
accuracy: 0.7368 - val_loss: 0.6231 - val_accuracy: 0.7942
Epoch 910/1000
9727/9727 [=====] - 1s 60us/sample - loss: 0.7521 -
accuracy: 0.7357 - val_loss: 0.6226 - val_accuracy: 0.7939
Epoch 911/1000
9727/9727 [=====] - 1s 66us/sample - loss: 0.7626 -
accuracy: 0.7369 - val_loss: 0.6223 - val_accuracy: 0.7939
Epoch 912/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7590 -
```

```
accuracy: 0.7384 - val_loss: 0.6219 - val_accuracy: 0.7940
Epoch 913/1000
9727/9727 [=====] - 1s 60us/sample - loss: 0.7524 -
accuracy: 0.7364 - val_loss: 0.6213 - val_accuracy: 0.7943
Epoch 914/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7585 -
accuracy: 0.7340 - val_loss: 0.6209 - val_accuracy: 0.7946
Epoch 915/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7510 -
accuracy: 0.7374 - val_loss: 0.6203 - val_accuracy: 0.7949
Epoch 916/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7560 -
accuracy: 0.7326 - val_loss: 0.6199 - val_accuracy: 0.7949
Epoch 917/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7654 -
accuracy: 0.7368 - val_loss: 0.6198 - val_accuracy: 0.7953
Epoch 918/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7501 -
accuracy: 0.7397 - val_loss: 0.6194 - val_accuracy: 0.7945
Epoch 919/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7452 -
accuracy: 0.7434 - val_loss: 0.6189 - val_accuracy: 0.7948
Epoch 920/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7565 -
accuracy: 0.7360 - val_loss: 0.6184 - val_accuracy: 0.7945
Epoch 921/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7502 -
accuracy: 0.7387 - val_loss: 0.6183 - val_accuracy: 0.7942
Epoch 922/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7557 -
accuracy: 0.7402 - val_loss: 0.6180 - val_accuracy: 0.7943
Epoch 923/1000
9727/9727 [=====] - 1s 60us/sample - loss: 0.7521 -
accuracy: 0.7355 - val_loss: 0.6176 - val_accuracy: 0.7946
Epoch 924/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7521 -
accuracy: 0.7377 - val_loss: 0.6171 - val_accuracy: 0.7949
Epoch 925/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7427 -
accuracy: 0.7401 - val_loss: 0.6167 - val_accuracy: 0.7951
Epoch 926/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7608 -
accuracy: 0.7362 - val_loss: 0.6166 - val_accuracy: 0.7957
Epoch 927/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7382 -
accuracy: 0.7421 - val_loss: 0.6161 - val_accuracy: 0.7954
Epoch 928/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7483 -
accuracy: 0.7395 - val_loss: 0.6158 - val_accuracy: 0.7956
Epoch 929/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7487 -
accuracy: 0.7415 - val_loss: 0.6157 - val_accuracy: 0.7960
Epoch 930/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7348 -
accuracy: 0.7430 - val_loss: 0.6154 - val_accuracy: 0.7954
Epoch 931/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7488 -
```

```
accuracy: 0.7424 - val_loss: 0.6150 - val_accuracy: 0.7954
Epoch 932/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.7476 -
accuracy: 0.7447 - val_loss: 0.6148 - val_accuracy: 0.7956
Epoch 933/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7493 -
accuracy: 0.7413 - val_loss: 0.6146 - val_accuracy: 0.7957
Epoch 934/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7447 -
accuracy: 0.7414 - val_loss: 0.6139 - val_accuracy: 0.7959
Epoch 935/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7354 -
accuracy: 0.7465 - val_loss: 0.6138 - val_accuracy: 0.7957
Epoch 936/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7371 -
accuracy: 0.7452 - val_loss: 0.6134 - val_accuracy: 0.7962
Epoch 937/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7470 -
accuracy: 0.7452 - val_loss: 0.6132 - val_accuracy: 0.7956
Epoch 938/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.7368 -
accuracy: 0.7460 - val_loss: 0.6131 - val_accuracy: 0.7957
Epoch 939/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7464 -
accuracy: 0.7355 - val_loss: 0.6127 - val_accuracy: 0.7962
Epoch 940/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7377 -
accuracy: 0.7441 - val_loss: 0.6126 - val_accuracy: 0.7960
Epoch 941/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.7403 -
accuracy: 0.7377 - val_loss: 0.6123 - val_accuracy: 0.7965
Epoch 942/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7404 -
accuracy: 0.7431 - val_loss: 0.6118 - val_accuracy: 0.7968
Epoch 943/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7406 -
accuracy: 0.7443 - val_loss: 0.6112 - val_accuracy: 0.7968
Epoch 944/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7327 -
accuracy: 0.7468 - val_loss: 0.6104 - val_accuracy: 0.7969
Epoch 945/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7361 -
accuracy: 0.7463 - val_loss: 0.6100 - val_accuracy: 0.7979
Epoch 946/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7385 -
accuracy: 0.7406 - val_loss: 0.6094 - val_accuracy: 0.7976
Epoch 947/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7424 -
accuracy: 0.7410 - val_loss: 0.6093 - val_accuracy: 0.7973
Epoch 948/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7355 -
accuracy: 0.7429 - val_loss: 0.6088 - val_accuracy: 0.7973
Epoch 949/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7326 -
accuracy: 0.7445 - val_loss: 0.6081 - val_accuracy: 0.7979
Epoch 950/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7424 -
```

```
accuracy: 0.7379 - val_loss: 0.6079 - val_accuracy: 0.7979
Epoch 951/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7378 -
accuracy: 0.7429 - val_loss: 0.6075 - val_accuracy: 0.7979
Epoch 952/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7383 -
accuracy: 0.7440 - val_loss: 0.6074 - val_accuracy: 0.7980
Epoch 953/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7325 -
accuracy: 0.7435 - val_loss: 0.6072 - val_accuracy: 0.7985
Epoch 954/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7365 -
accuracy: 0.7430 - val_loss: 0.6069 - val_accuracy: 0.7990
Epoch 955/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7358 -
accuracy: 0.7396 - val_loss: 0.6067 - val_accuracy: 0.7991
Epoch 956/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7276 -
accuracy: 0.7487 - val_loss: 0.6062 - val_accuracy: 0.7991
Epoch 957/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7372 -
accuracy: 0.7405 - val_loss: 0.6061 - val_accuracy: 0.8005
Epoch 958/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7363 -
accuracy: 0.7464 - val_loss: 0.6055 - val_accuracy: 0.7993
Epoch 959/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7287 -
accuracy: 0.7494 - val_loss: 0.6053 - val_accuracy: 0.7996
Epoch 960/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7285 -
accuracy: 0.7486 - val_loss: 0.6048 - val_accuracy: 0.7993
Epoch 961/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7295 -
accuracy: 0.7462 - val_loss: 0.6043 - val_accuracy: 0.7988
Epoch 962/1000
9727/9727 [=====] - 1s 63us/sample - loss: 0.7369 -
accuracy: 0.7435 - val_loss: 0.6039 - val_accuracy: 0.7994
Epoch 963/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7270 -
accuracy: 0.7481 - val_loss: 0.6036 - val_accuracy: 0.7999
Epoch 964/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7337 -
accuracy: 0.7443 - val_loss: 0.6036 - val_accuracy: 0.8008
Epoch 965/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7296 -
accuracy: 0.7489 - val_loss: 0.6035 - val_accuracy: 0.8008
Epoch 966/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7280 -
accuracy: 0.7476 - val_loss: 0.6032 - val_accuracy: 0.8011
Epoch 967/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7158 -
accuracy: 0.7509 - val_loss: 0.6028 - val_accuracy: 0.8014
Epoch 968/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7328 -
accuracy: 0.7398 - val_loss: 0.6024 - val_accuracy: 0.8008
Epoch 969/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7346 -
```

```
accuracy: 0.7470 - val_loss: 0.6022 - val_accuracy: 0.8006
Epoch 970/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7339 -
accuracy: 0.7479 - val_loss: 0.6020 - val_accuracy: 0.8005
Epoch 971/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7262 -
accuracy: 0.7476 - val_loss: 0.6016 - val_accuracy: 0.8011
Epoch 972/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7291 -
accuracy: 0.7496 - val_loss: 0.6011 - val_accuracy: 0.8008
Epoch 973/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7205 -
accuracy: 0.7477 - val_loss: 0.6007 - val_accuracy: 0.8002
Epoch 974/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7213 -
accuracy: 0.7473 - val_loss: 0.6003 - val_accuracy: 0.8002
Epoch 975/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7212 -
accuracy: 0.7474 - val_loss: 0.6001 - val_accuracy: 0.8010
Epoch 976/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7335 -
accuracy: 0.7447 - val_loss: 0.5998 - val_accuracy: 0.8013
Epoch 977/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7247 -
accuracy: 0.7510 - val_loss: 0.5990 - val_accuracy: 0.8003
Epoch 978/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7273 -
accuracy: 0.7467 - val_loss: 0.5988 - val_accuracy: 0.8005
Epoch 979/1000
9727/9727 [=====] - 1s 54us/sample - loss: 0.7199 -
accuracy: 0.7536 - val_loss: 0.5986 - val_accuracy: 0.8003
Epoch 980/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7193 -
accuracy: 0.7501 - val_loss: 0.5982 - val_accuracy: 0.8008
Epoch 981/1000
9727/9727 [=====] - 1s 53us/sample - loss: 0.7187 -
accuracy: 0.7558 - val_loss: 0.5978 - val_accuracy: 0.8008
Epoch 982/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.7174 -
accuracy: 0.7516 - val_loss: 0.5973 - val_accuracy: 0.8010
Epoch 983/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.7206 -
accuracy: 0.7504 - val_loss: 0.5968 - val_accuracy: 0.8003
Epoch 984/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7190 -
accuracy: 0.7483 - val_loss: 0.5965 - val_accuracy: 0.8008
Epoch 985/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7286 -
accuracy: 0.7498 - val_loss: 0.5963 - val_accuracy: 0.8006
Epoch 986/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7099 -
accuracy: 0.7575 - val_loss: 0.5958 - val_accuracy: 0.8010
Epoch 987/1000
9727/9727 [=====] - 1s 56us/sample - loss: 0.7228 -
accuracy: 0.7487 - val_loss: 0.5953 - val_accuracy: 0.8003
Epoch 988/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.7172 -
```

```

    accuracy: 0.7507 - val_loss: 0.5951 - val_accuracy: 0.8011
Epoch 989/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.7238 -
    accuracy: 0.7558 - val_loss: 0.5946 - val_accuracy: 0.8013
Epoch 990/1000
9727/9727 [=====] - 1s 55us/sample - loss: 0.7182 -
    accuracy: 0.7502 - val_loss: 0.5943 - val_accuracy: 0.8010
Epoch 991/1000
9727/9727 [=====] - 1s 59us/sample - loss: 0.7158 -
    accuracy: 0.7503 - val_loss: 0.5938 - val_accuracy: 0.8014
Epoch 992/1000
9727/9727 [=====] - 1s 75us/sample - loss: 0.7188 -
    accuracy: 0.7544 - val_loss: 0.5935 - val_accuracy: 0.8010
Epoch 993/1000
9727/9727 [=====] - 1s 59us/sample - loss: 0.7200 -
    accuracy: 0.7509 - val_loss: 0.5933 - val_accuracy: 0.8010
Epoch 994/1000
9727/9727 [=====] - 1s 58us/sample - loss: 0.7172 -
    accuracy: 0.7496 - val_loss: 0.5931 - val_accuracy: 0.8017
Epoch 995/1000
9727/9727 [=====] - 1s 57us/sample - loss: 0.7127 -
    accuracy: 0.7537 - val_loss: 0.5927 - val_accuracy: 0.8011
Epoch 996/1000
9727/9727 [=====] - 1s 65us/sample - loss: 0.7208 -
    accuracy: 0.7569 - val_loss: 0.5925 - val_accuracy: 0.8006
Epoch 997/1000
9727/9727 [=====] - 1s 71us/sample - loss: 0.7330 -
    accuracy: 0.7467 - val_loss: 0.5923 - val_accuracy: 0.8010
Epoch 998/1000
9727/9727 [=====] - 1s 82us/sample - loss: 0.7148 -
    accuracy: 0.7518 - val_loss: 0.5921 - val_accuracy: 0.8013
Epoch 999/1000
9727/9727 [=====] - 1s 78us/sample - loss: 0.7145 -
    accuracy: 0.7488 - val_loss: 0.5917 - val_accuracy: 0.8013
Epoch 1000/1000
9727/9727 [=====] - 1s 88us/sample - loss: 0.7068 -
    accuracy: 0.7552 - val_loss: 0.5912 - val_accuracy: 0.8013

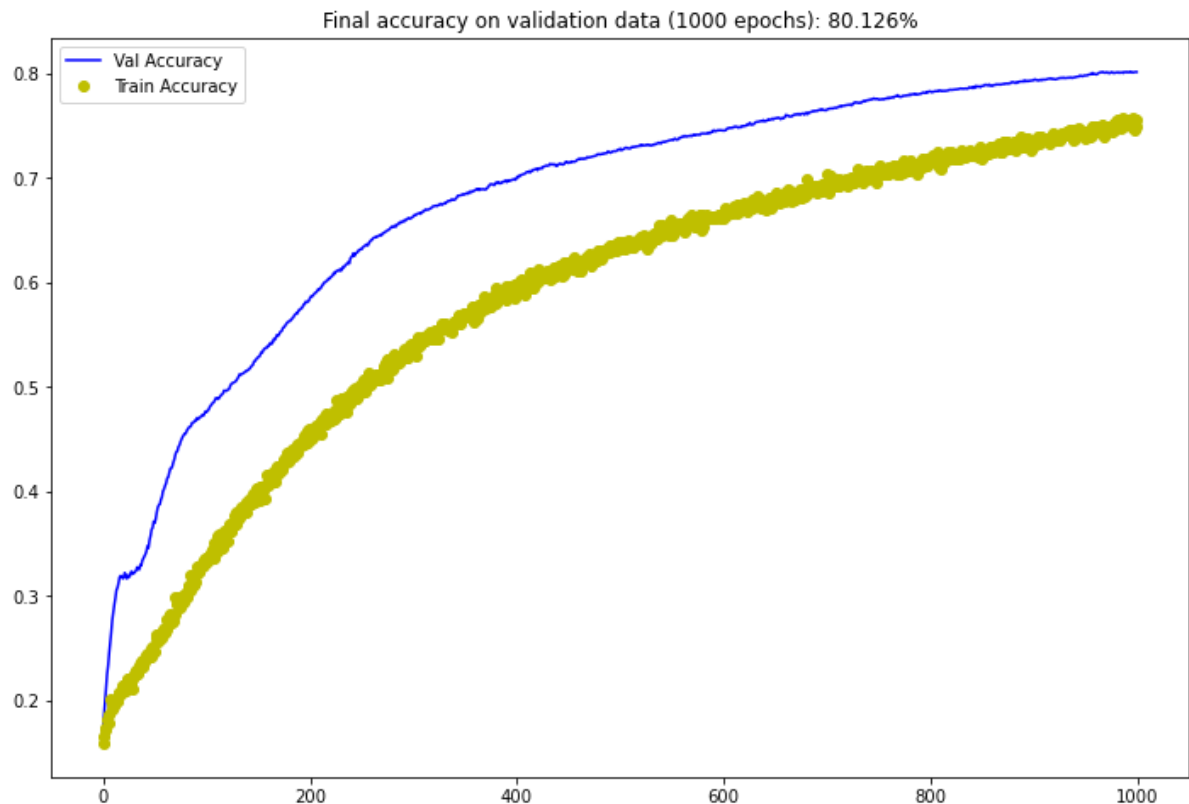
```

```
In [7]: hist.history.keys()
```

```
Out[7]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
In [24]: import matplotlib.pyplot as plt
%matplotlib inline

plt.figure(figsize=(12,8))
plt.plot(hist.history['val_accuracy'], 'b-', label='Val Accuracy')
plt.plot(hist.history['accuracy'], 'yo', label='Train Accuracy')
plt.title("Final accuracy on validation data (1000 epochs): {:.3%}".format(hist.history['val_accuracy'][-1:][0]))
plt.legend(loc=2)
_ = plt.show()
```



Make Prediction

Let us see whether trained network could properly classify sounds for bed, cat, happy. Test with recorded sounds in the test data set.


```
In [25]: # `bed` with southern accent
# saved vectors bed, cat, happy
# '/happy/0685264e_nohash_0.wav'
# 'd486fb84_nohash_0.wav, happy d486fb84_nohash_1.wav'
# cat 6ac35824_nohash_0.wav, 6ac35824_nohash_1.wav
train_audio_path = './data/'
filename = 'bed/004ae714_nohash_1.wav'
new_sample_rate = 16000
samples, sample_rate = librosa.load(str(train_audio_path) + filename)
ipd.Audio(samples, rate=sample_rate)
```

Out[25]:

0:00 / 0:01

```
In [27]: print(predict('./data/bed/004ae714_nohash_1.wav', model=model))

five
```

```
In [28]: train_audio_path = './data/'
filename = 'cat/6ac35824_nohash_0.wav'
new_sample_rate = 16000
samples, sample_rate = librosa.load(str(train_audio_path) + filename)
ipd.Audio(samples, rate=sample_rate)
```

Out[28]:

0:00 / 0:01

```
In [29]: print(predict('./data/cat/6ac35824_nohash_0.wav', model=model))

cat
```

```
In [30]: train_audio_path = './data/'
filename = 'happy/0685264e_nohash_0.wav'
new_sample_rate = 16000
samples, sample_rate = librosa.load(str(train_audio_path) + filename)
ipd.Audio(samples, rate=sample_rate)
```

Out[30]:

0:00 / 0:01

```
In [31]: print(predict('./data/happy/0685264e_nohash_0.wav', model=model))

happy
```

```
In [32]: train_audio_path = './data/'  
         #filename = 'yes/0a7c2a8d_nohash_0.wav'  
         filename = 'seven/b1114e4f_nohash_0.wav'  
         new_sample_rate = 16000  
         samples, sample_rate = librosa.load(str(train_audio_path) + filename)  
         ipd.Audio(samples, rate=sample_rate)
```

Out[32]:

0:00 / 0:01

```
In [33]: print(predict('./data/seven/b1114e4f_nohash_0.wav', model=model))  
five
```

Some predictions are good and some not so good.

In []: