INDEX for Project Documentation

INTRODUCTION

Project Profile

Project Title	cartoon-character automation using deep learning
Project Definition	A deep learning utility to recognise characters and generate scripts (dialogues)
Technology Used	Flask, Python (TensorFlow I Keras), HTML, CSS, JavaScript and JQuery
Front-end	HTML, CSS, JavaScript, Bootstrap, and JQuery, Tensorboard (Visualization)
Back-end	Flask, TensorFlow and Keras
Internal Guide	Yesha ma’am
Documentation Generation Tool	Google Docs
Created By	Kush Gabani

SYSTEM INTRODUCTION

System Definition

- A web application to automate cartoon-characters processes for industrial usage for the famous ***American TV Series "The Simpsons"*** with deep learning concepts.

- The application consists of two modules - Character Recognition and Script Generation. The Character Recognition module recognises and classifies images from the wide variety of characters in the series, 20 characters to be specific; with the help of ***Deep Convolutional Neural Networks (DCNN)*** with the relatively high accuracy of *99.7%.*
  Character-Recognition Module.

- The Script Generation module generates **n** number of new dialogues for the episode given a few words as a context and beginning; with ***sequence-to-vector*** models consisting of ***Long Short Term Memory (LSTM) Cells,*** summing all up, the training phase resulted in a low Mean Absolute Error (MAE) of *0.03*
  Script-Generation Module

- Model architecture, weight distribution and hyperparameters visualization is made public to the user, so they can access and analyze the deep learning models as and when needed to understand much better.

System Objectives

- The project aims at automating various processes in industrial areas like analysis and even on a contextual level.
- The Character Recognition module should be able to accurately classify among the 20 characters using images as an input.
- The Script Generation module is capable of generating as many new dialogues as needed given a context of around 10-20 words
- Model architectures and visualization of both the trained deep learning can be used for analysis of the training phase of the models. The recorded log file data visualisation can be also used for hyper-parameter tuning to further improve the model's accuracy and decrease the loss or educational purposes like how the model trains? Or analysing the model's architecture.

System Scope

- The Character Recognition module can be modified into a meta-application or an API for another application like calculating “screen time (appearance)” of characters in each episode.

- The DCNN of the Character Recognition can also be used as a base model for Faster RCNN (Regional Convolutional Neural Network) in Object Detection application. Image segmentation can also be implemented using this DCNN as a base model. Furthermore, The same DCNN architecture can be modified slightly to be used as a high accuracy discriminator in GANs (General Adversarial Networks) to generate new Characters.

- The Script Generation module can further be fine-tuned to generate an entire new series with new characters replacing the old ones for example, The Simpsons but in Hindi using Google Translate API. The generated script can be further be used as an input sequence for Sentiment-Analysis application that can then classify the emotion of the generated script.

- Model Architecture Analysis and understanding the deep neural networks at a relatively low-level. This can be used as a stage to then analyze a model and write a research, thesis or dissertation on these models.

Hardware & Software Requirements

Hardware Requirements

- Training Phase (Google Colab) -

  PROCESSOR - Xeon Processors @2.3Ghz
  RAM - 12GB DDR5 RAM
  SSD - 64GB
  GPU - Tesla K80 having 2496 CUDA cores
  RUNTIME ENV. - 12 Hours (Total)

- Deployment Phase (Personal Device) -

  PROCESSOR - Intel i5 Processor @1.4Ghz
  RAM - 4GB DDR3 RAM
  SSD - 64 GB

Software Requirements

- Operating System - Windows 7 or MacOS Cataline and above
- Training Environment - Google Colab with GPU (https://colab.research.google.com/)
- Front-end Tools - HTML, CSS, JavaScript (best works on Chrome's V8 engine)
- Back-end Tools - Flask, TensorFlow (v2.x>=), Keras

REQUIREMENT ANALYSIS AND MODELING

Expected working of the system

- **Home Page**
  - This is the default page that first loads when the flask server is in deployment. When run locally, the web application can be accessed with Localhost PORT number 5000 as http://localhost:5000/
  - The home page will define the route to the two main modules of the project; namely, Character Recognition and Script Generation.
  - When clicked on either of them, the web application is redirected to that specific module where the deployed model functions.
  - Code for both the modules can be accessed by clicking on the GitHub icons on either of the modules.

- **Character Recognition Page**
  - The webpage where the DCNN model for Character Recognition is deployed. The user should be able to upload any cartoon image that needs to be classified among the 20 characters.
  - The image is then uploaded to the server using JQuery and AJAX. To be able to pass the image to the DCNN, it first needs to be pre-processed. The image is read into an nd-array then resized to 100 x 100 resolution.
  - The DCNN then should be able to classify and display the output on the web page accurately.
  - The user can also visualize the training phase of the DCNN Model and its architecture, analyze the weight distribution of internal parameters and tune hyper-parameters for educational purposes.
  - Model Statistics of the DCNN should also be viewed by the user for reference.

- **Script Generation Page**
  - The webpage where the Sequence-to-Vector LSTM Model for Script Generation is deployed. The user should be able to input the number of words (default = 100) to be generated by the Neural Network and a few

words as a context for the dialogues to be generated, generally 30-40 words.

- If the user does not want to enter any context then three default contexts are provided for the input.

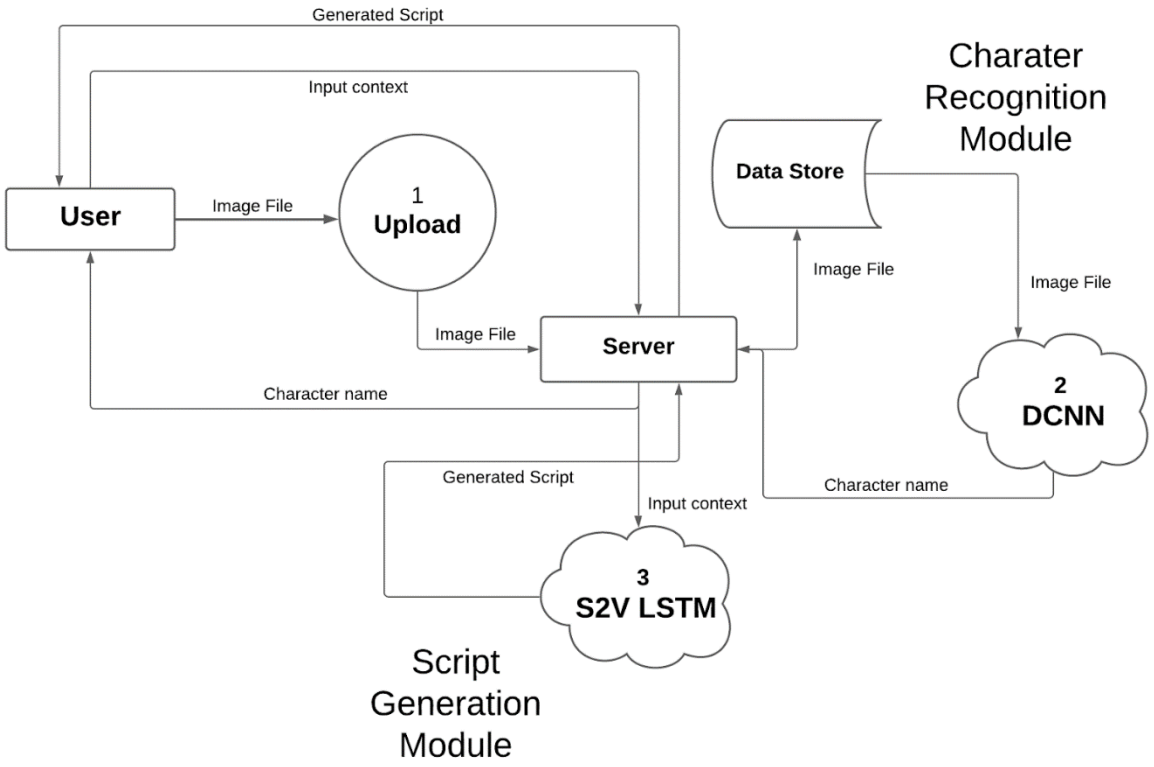
- The user should also be able to visualize the training phase of the Sequence-to-Vector LSTM model in graphs, flowcharts and tune hyper-parameters. This should also be enough for a user to base their dissertation on.

- Model Statistics of the deployed model should also be accessible to the user.

- The user should also be able to download the generated text into a text file for their personal future use.

Data flow diagrams (DFD)

DFD Level 0



Process Specification

Process 1

Process Name	Upload
Entity	Users
Data Store	Local data store on server
Input	Image File
Output	Image uploaded on server
Description	The image uploaded by the user for Character Recognition is stored on the server for further use.

Process 2

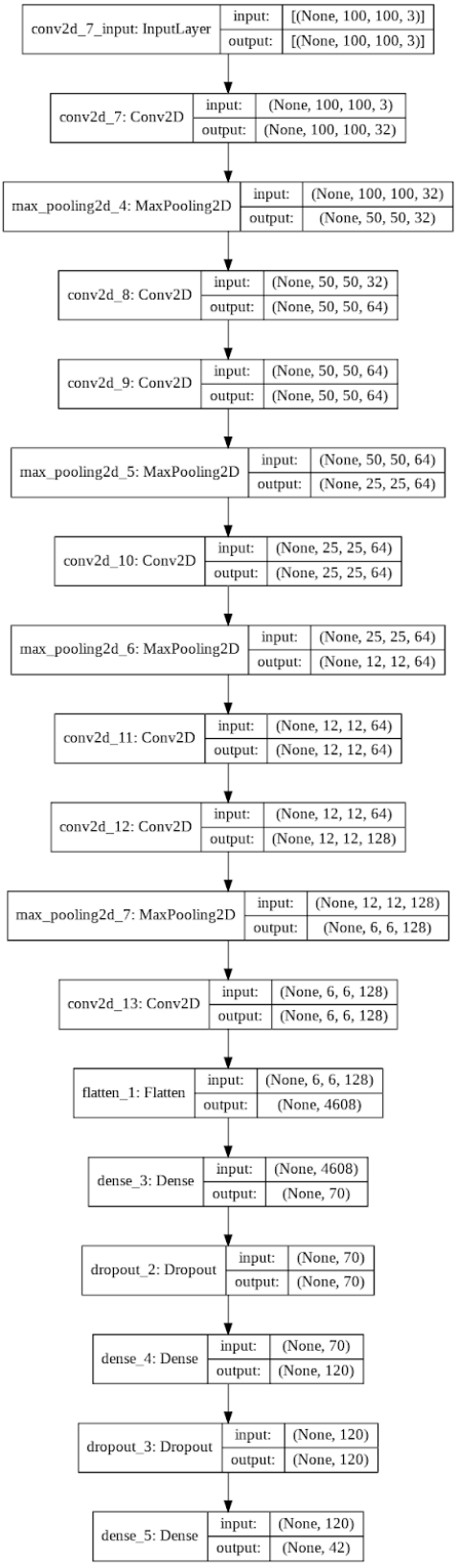
Process Name	DCNN
Entity	Data Store, DCNN model
Input	Image uploaded on the server
Output	Classified character name
Description	The image uploaded by the user is pre-processed and passed as input to the DCNN to classify and predict and character

Process 3

Process Name	S2V LSTM
Entity	Server, S2V LSTM
Input	Context input from the user
Output	Sequence of generated dialogues
Description	The context input is pre-processed and passed to LSTM model to generate script

Model Architecture

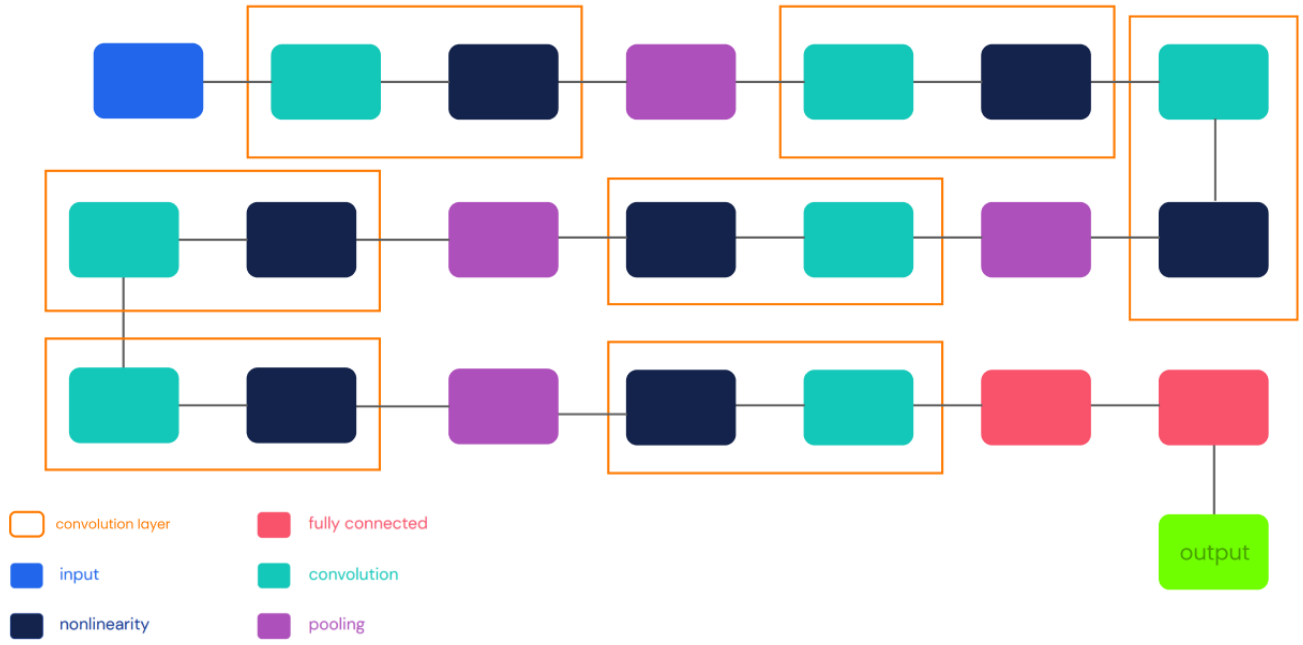
Character Recognition's DCNN Model Architecture



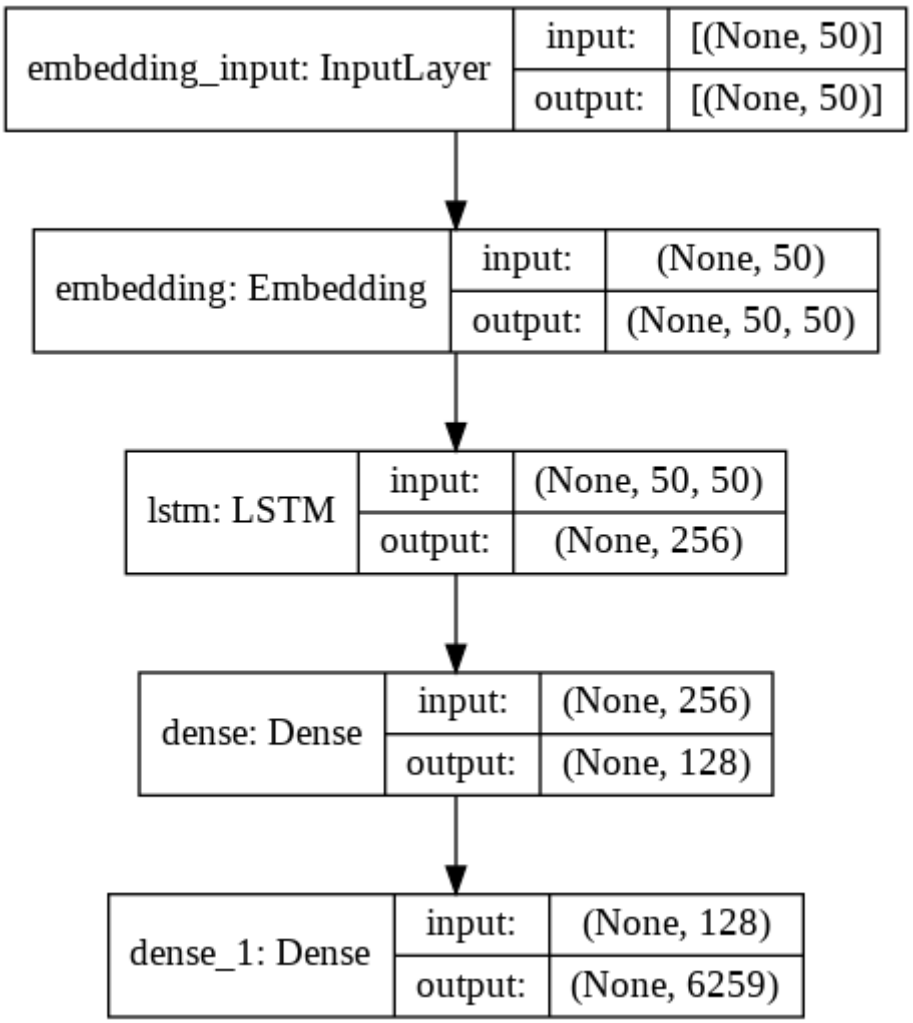
The image once uploaded by the user, is stored on the server. The model then reads the image in a NumPy array and then converts from BGR format to RGB format. The colour

corrected image format is then resized into a 100 x 100 image to pass it to the main DCNN model as an Input Layer. The DCNN architecture is made up of 7 Convolutional Layers divided into 4 Max Pooling Layers. Once the features are collected by passing the pre-processed image, they are converted into a one-dimensional vector with a Flatten layer. This sequence of flattened pixels is then passed on to a Dense feed-forward Neural Networks with a Dropout layer after each hidden layer. The Output Layer consists of 42 neurons each for a character to be recognised. All the Dense layers uses a “ReLu” activation except the Output Layer that uses a “Softmax” activation.
Adam is the optimizer and Categorical Cross-Entropy is the Loss function used for Backpropagation to train the DCNN for 30 epochs.
During the training phase, Data Augmentation is performed on the images using a Random Over Sampler that helps to increase the number of images to train on. Resulting in a dynamically trained model that is good at classifying characters. The first base model was trained on this augmented dataset for 20 epochs and a batch size of 32. This model reached an accuracy of 90.3%, then further hyper-parameter tuning was Multiple times to find the best accuracy the model can reach. After tuning the batch size, the number of epochs, the optimizer used, the dropout rate using Grid Search CV and Random Search CV from Sci-kit learn wrapper class integrated with the Keras model architecture, the model reached the accuracy of 99.4%

Character Recognition module’s DCNN Architecture



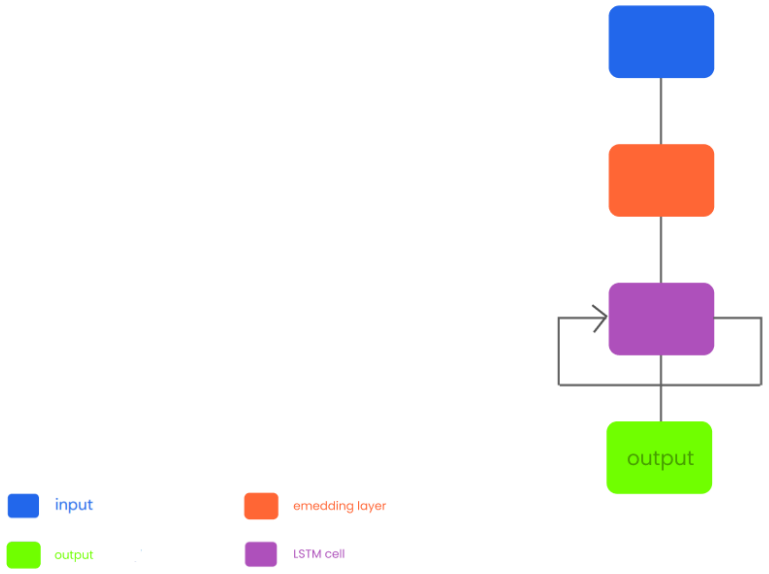
Script Generation's Sequence-to-Vector LSTM Model Architecture



The user inputs context sentences based on which dialogues are generated by the model. The context input is first converted into a sequence of integers encoded by a Tokenizer fit on the training set. The integer sequence is then padded from left if the number of words in the context are less than 50 and if they are more than 50 then the first few words are removed in order to make the sequence of exactly 50 words. This sequence of 50 integers is then passed into the Sequence-to-Vector LSTM model. The Model is made up of the Input Layer passing the padded-context integer sequence to an Embedding Layer where they are transformed to a vector in 50-dimensional space. This embedded vector are then passed to a LSTM Layer containing 256 LSTM cells. The output from the LSTM layer is then passed to a Dense Layer with 128 neurons that uses "ReLu" activation function. The last layer is another Dense Layer called the Output layer that uses "Softmax" activation function, thus the Output Layers produces a probability distribution for 6259 words i.e., the vocabulary of the training set. Thus, each value in the probability distribution represents the probability of that word being the next word. The integer value with the highest probability is our final output for that context sequence. This Integer encoded value is then converted into its word-form using the same Tokenizer, this word is concatenated with the context sentence. The input context to generate the next word would be from second word to the new output generated thus removing the first word from input. And this whole process is repeated n times

where n is the number of words that needs to be generated that is also taken as an input by the user.
During the training phase of this Sequence-to-Vector model, Adam was used as an optimizer and Mean Absolute Error (MAE) as a Loss function. Alongside these, a batch size of 128, a sequence length of 50 and epochs were set to 180. This resulted in an MAE of 0.04.

Script Generation module's S2V LSTM Architecture



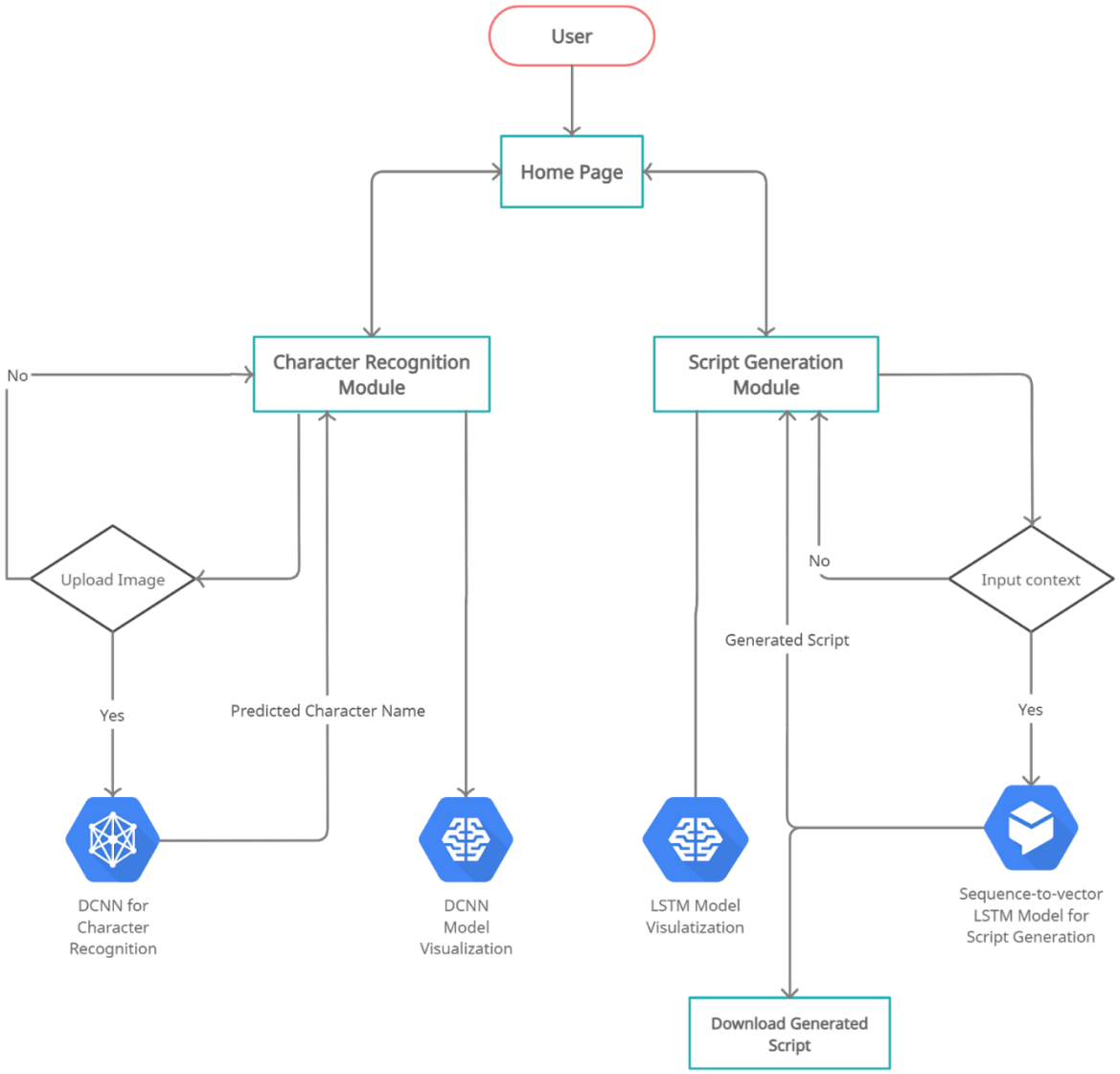
Data Dictionary (Training data)

- **Character Recognition**
  - 20,933 cartoon RBB images of characters from *The Simpsons* with each character having 400-1500 images of its own.
    Character Image Dataset

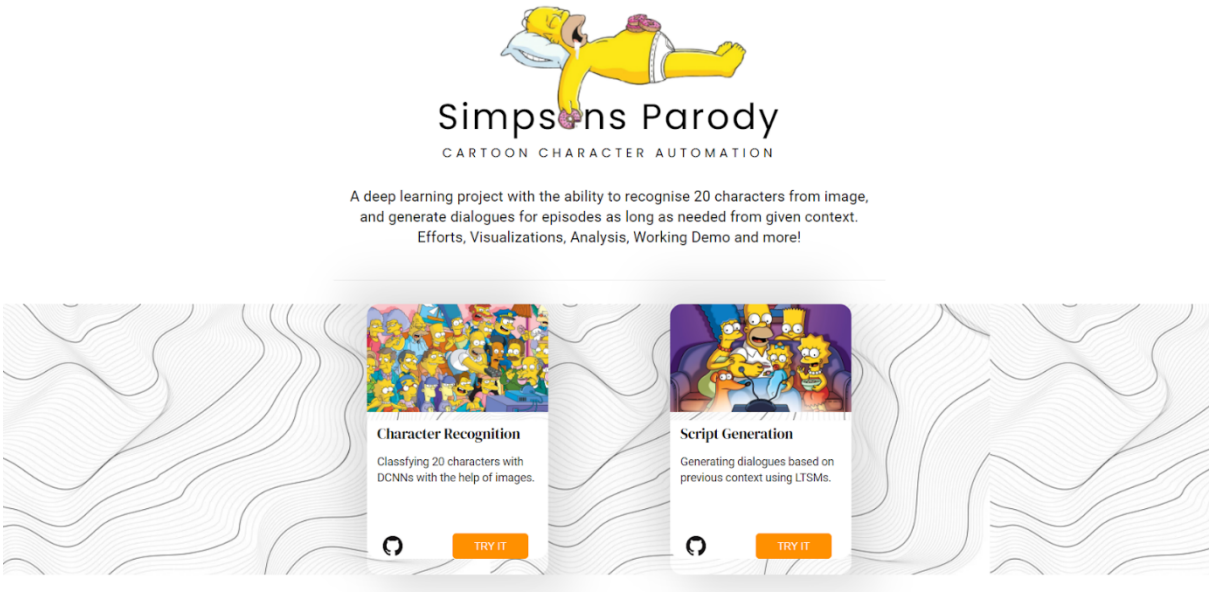
- **Script Generation**
  - Dialogues from the series *The Simpsons* consisting of around 80,000 words to process and generate more than 30,00,000 sequences to train the Sequence-to-Vector LSTM model on

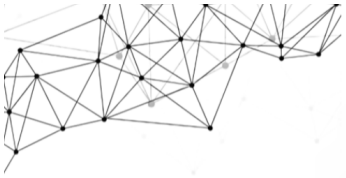
DESIGN

System Flowchart



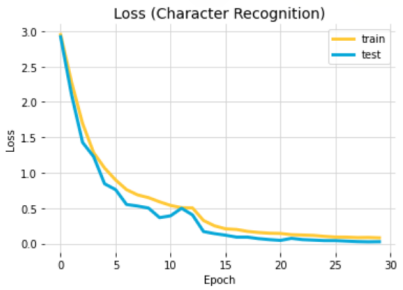
Design Screenshots

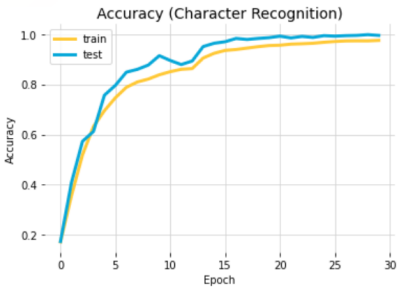




Model Statistics

0.039	99.3%	0.99
Loss	Accuracy	ROC AUC curve

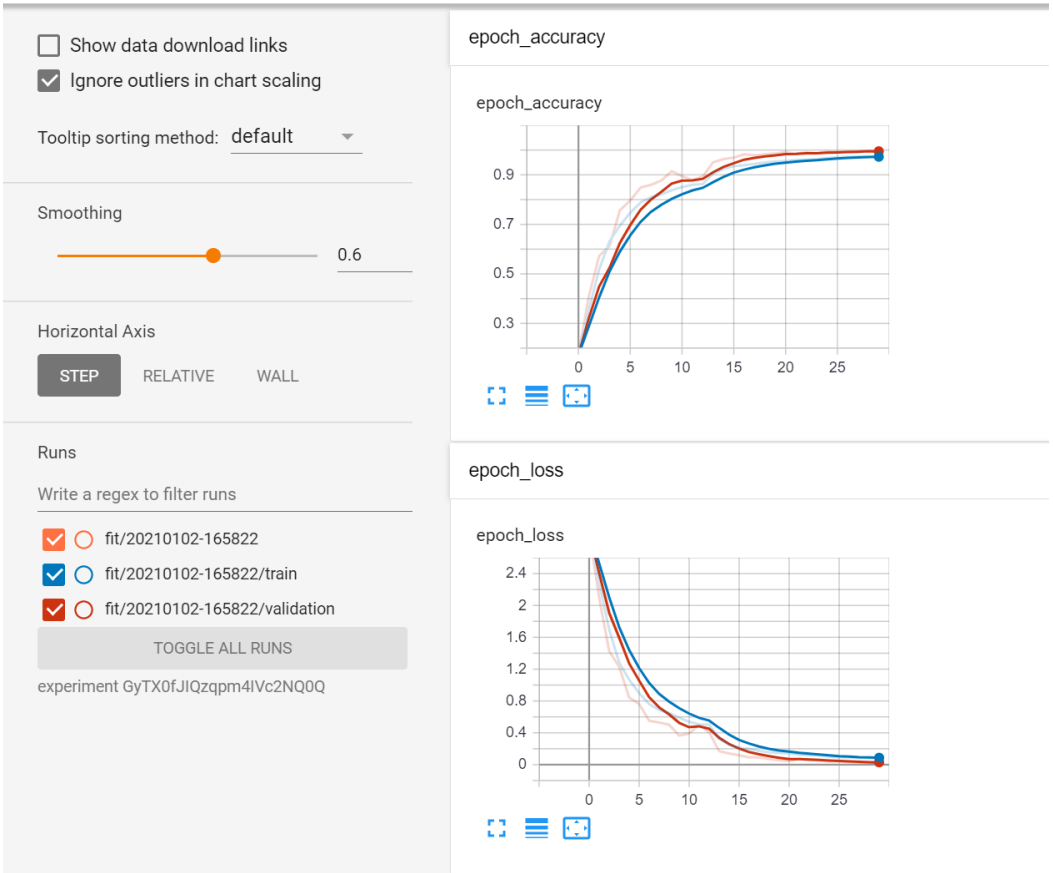


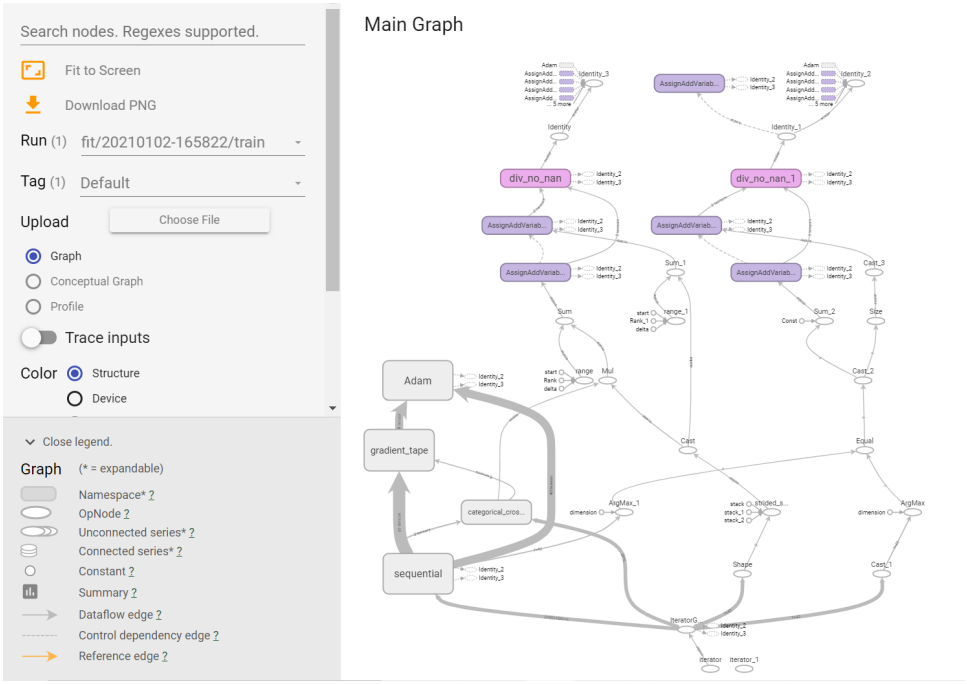


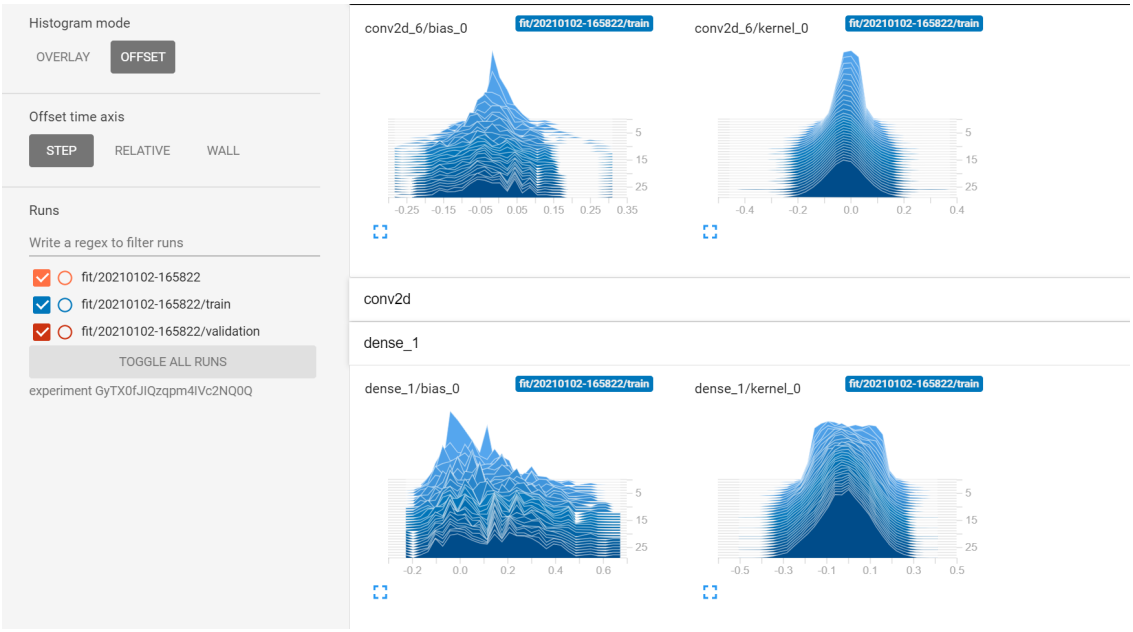
Visualize

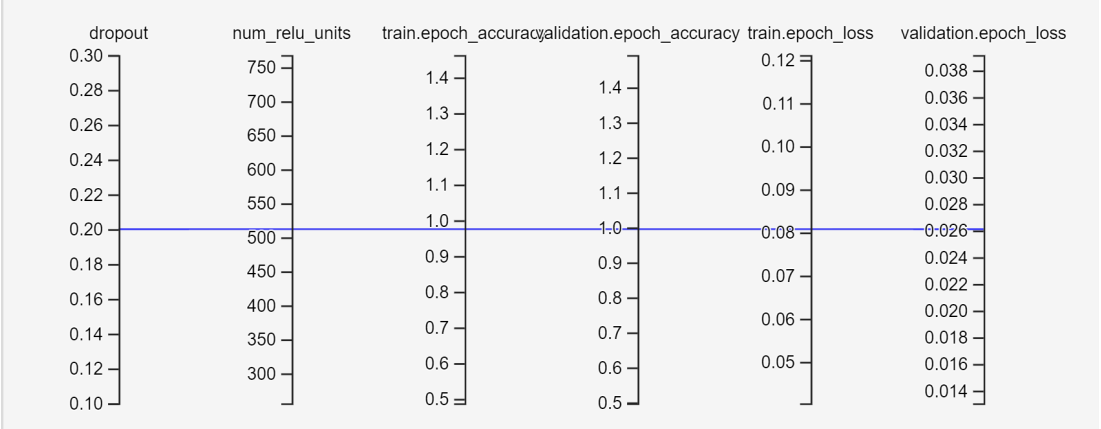
Simpsons Character Recognition using Deep Convolutional Neural Networks

Training results from https://colab.research.google.com/drive/11KYv4B26DVD5YIzhzuvtZV3NnEcR0pmN#scrollTo=kHe3koM4DeOY

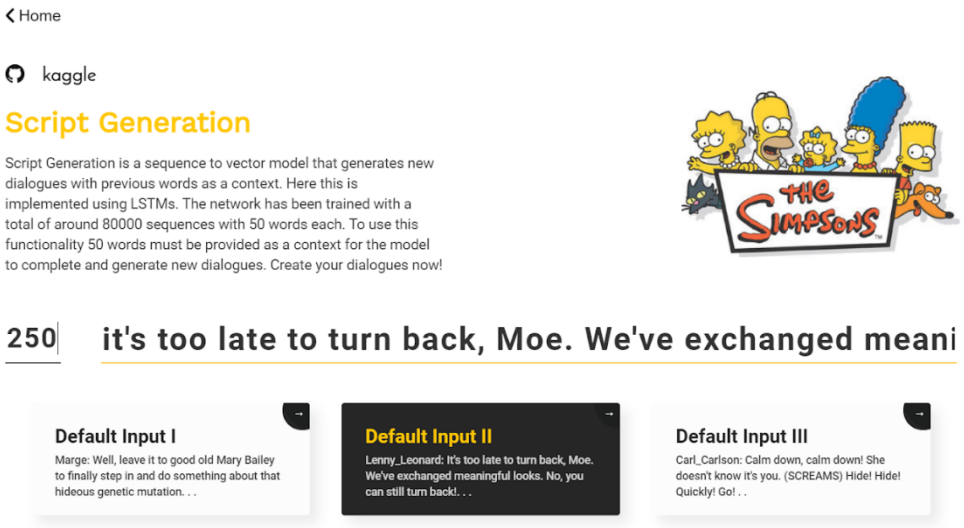



Search nodes. Regexes supported.
Fit to Screen
Download PNG
Run (1) fit/20210102-165822/train
Tag (1) Default
Upload Choose File
Graph
Conceptual Graph
Profile
Trace inputs
Color Structure
Device
Close legend.
Graph (* = expandable)
Namespace* ?
OpNode ?
Unconnected series* ?
Connected series* ?
Constant ?
Summary ?
Dataflow edge ?
Control dependency edge ?
Reference edge ?
Main Graph
Adam
gradient_tape
sequential
div_no_nan
div_no_nan_1


Histogram mode
OVERLAY
OFFSET
Offset time axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
fit/20210102-165822
fit/20210102-165822/train
fit/20210102-165822/validation
TOGGLE ALL RUNS
experiment GyTX0fJIQzqpm4IVc2NQ0Q
conv2d_6/bias_0
fit/20210102-165822/train
conv2d_6/kernel_0
fit/20210102-165822/train
conv2d
dense_1
dense_1/bias_0
fit/20210102-165822/train
dense_1/kernel_0
fit/20210102-165822/train


dropout
num_relu_units
train.epoch_accuracy
validation.epoch_accuracy
train.epoch_loss
validation.epoch_loss

DESIGN


Home
kaggle
Script Generation
Script Generation is a sequence to vector model that generates new dialogues with previous words as a context. Here this is implemented using LSTMs. The network has been trained with a total of around 80000 sequences with 50 words each. To use this functionality 50 words must be provided as a context for the model to complete and generate new dialogues. Create your dialogues now!
250
it's too late to turn back, Moe. We've exchanged meani
Default Input I
Marge: Well, leave it to good old Mary Bailey to finally step in and do something about that hideous genetic mutation. . .
Default Input II
Lenny_Leonard: It's too late to turn back, Moe. We've exchanged meaningful looks. No, you can still turn back!. . .
Default Input III
Carl_Carlson: Calm down, calm down! She doesn't know it's you. (SCREAMS) Hide! Hide! Quickly! Go! . .

Lenny_Leonard: It's too late to turn back, Moe. We've exchanged meaningful looks.
Moe_Szyslak: No, you can still turn back! The point of no return is the whispered huddle.
Moe_Szyslak: Oh God, oh God!
Homer_Simpson: Ahhh, that is so much better than hospital beer.
Lenny_Leonard: Homer, where you been the last
one of night .
moe szyslak : aw , that ' s so hard on that ' - - he ' s just pointing to do with the fan who was hoping to do with the guy like on a sport .
rev .
hooper : well , i was meanin ' to drink to make the ladies .
here play your women flying of his twin years but then they wanted clothes by item .
duffman : if duff s stuff babe .
duffman : if only .
.
.
wait to my shows .
coach : i think that .
mrs .
powers : (disgusted) this is , of you .
stillwater : we ' re tellin ' ya , so i would just .
i ' m the designated back ! kemi : (mid

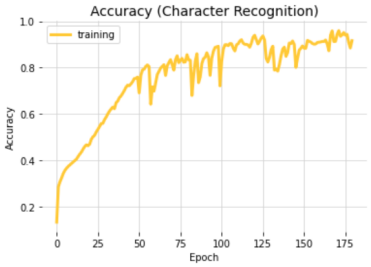
Download

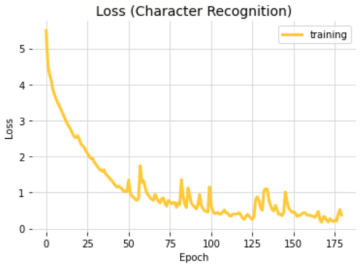
Model Statistics

0.098
Loss

97.1%
Accuracy

0.04
MAE

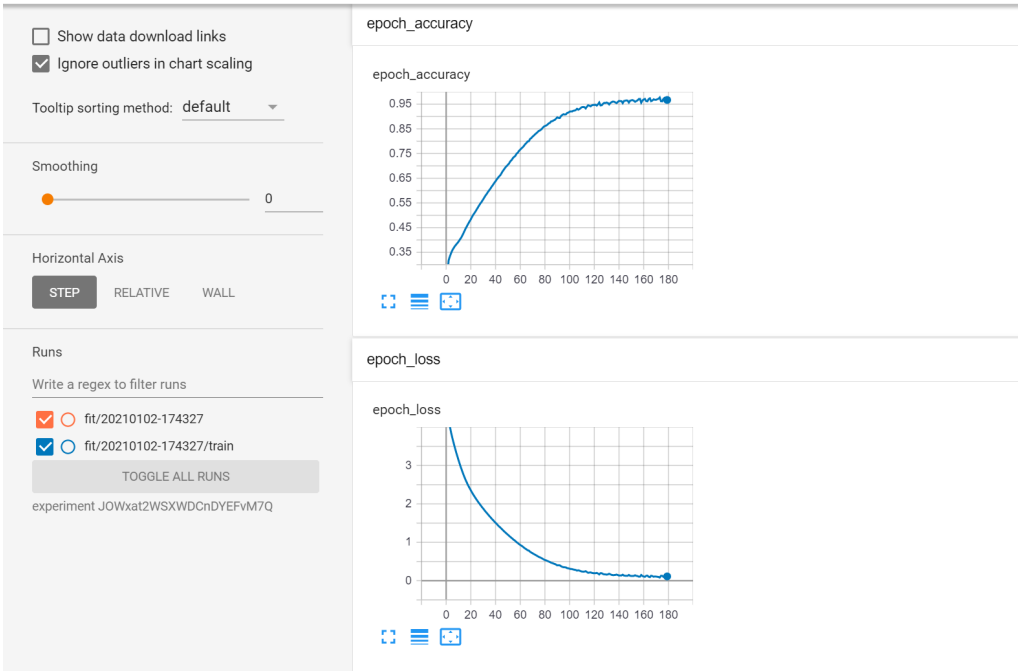




Visualize

Simpsons Script Generation using LSTM architecture

Training results from https://colab.research.google.com/drive/1RuolxiuHm60Xm9C-X4OtcT5scWTtwFWQ#scrollTo=FJO8Dn0oM25T

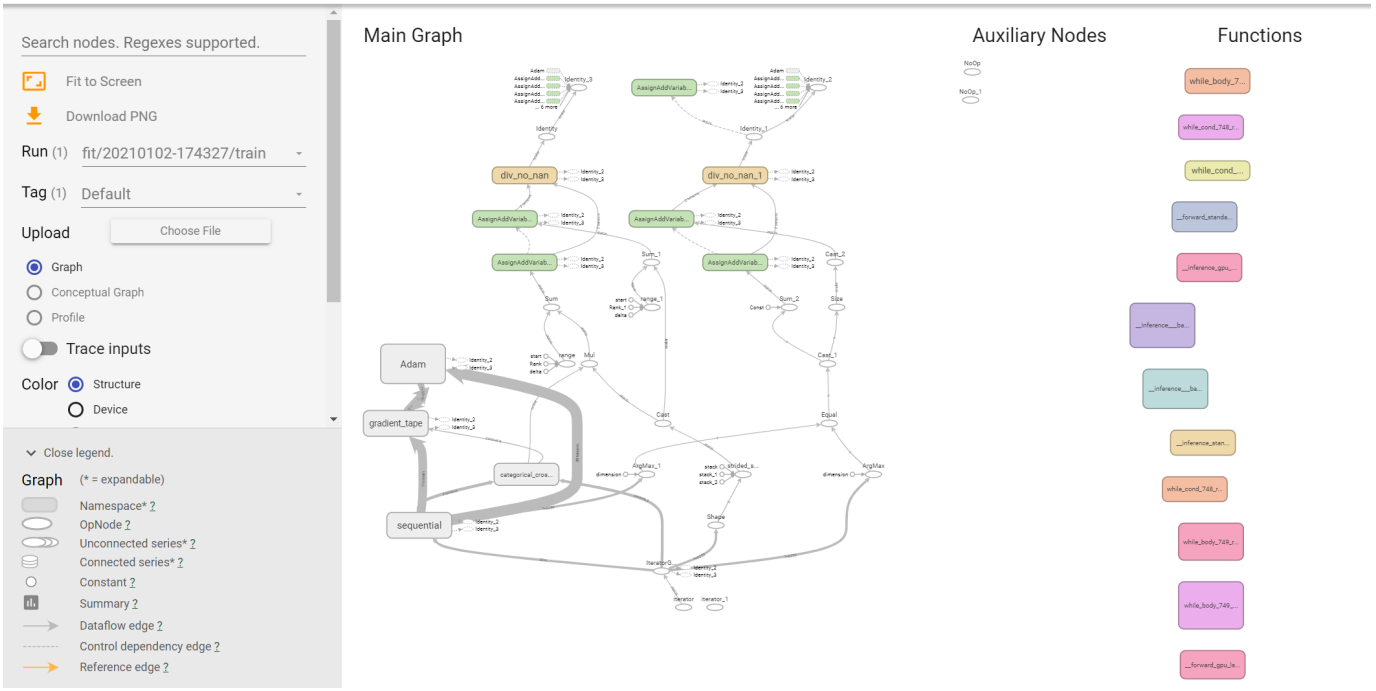


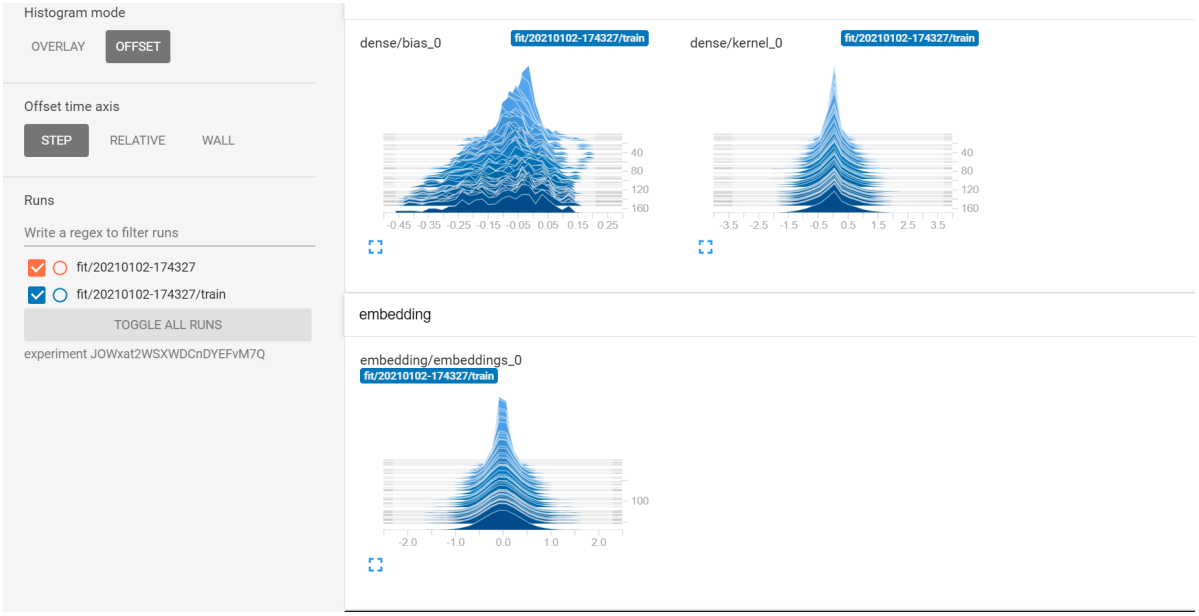
DESIGN

Simpsons Script Generation using LSTM architecture

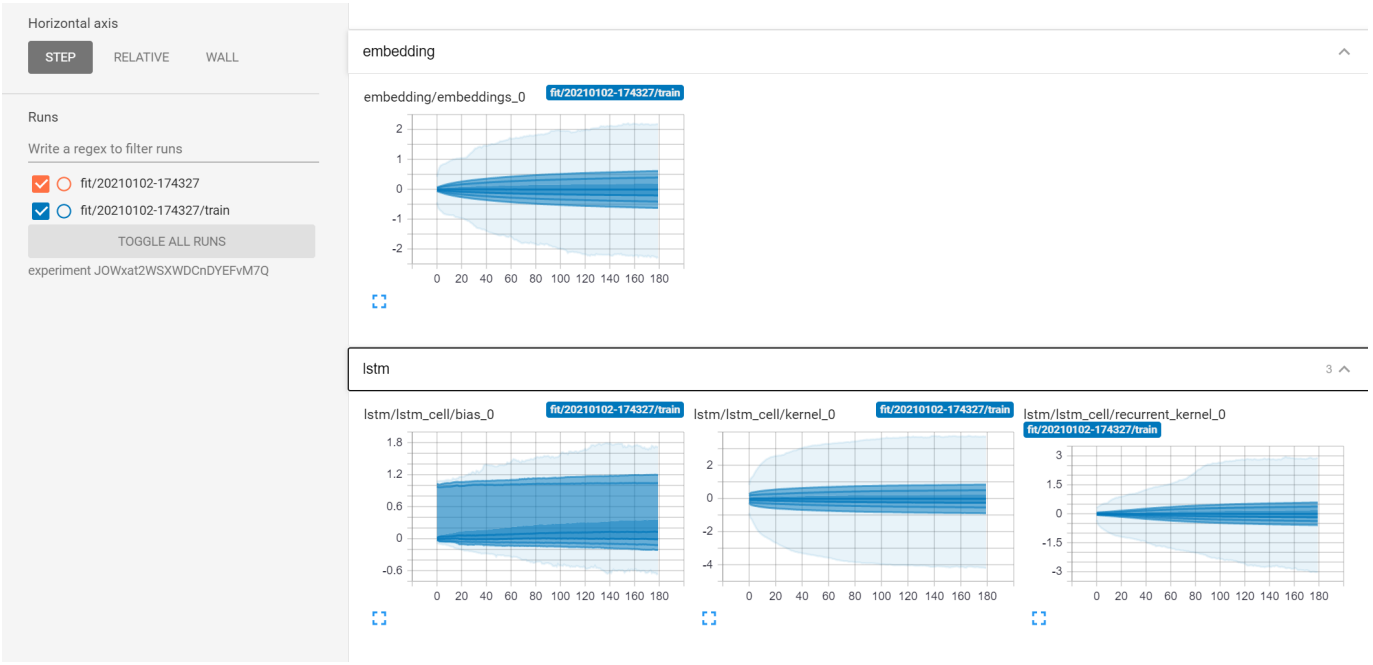
Created on Jan 2, 2021, 11:55:25 Pi

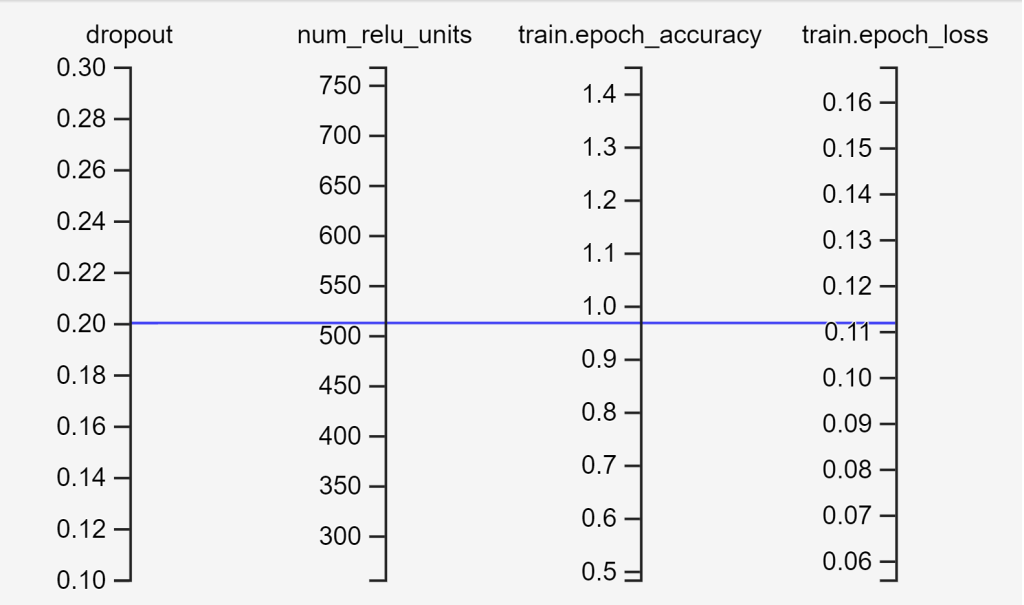
Training results from https://colab.research.google.com/drive/1RuolxiuHm60Xm9C-X4OtcT5scWTtwFWQ#scrollTo=FJO8Dn0oM25T





DESIGN


Horizontal axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
fit/20210102-174327
fit/20210102-174327/train
TOGGLE ALL RUNS
experiment JOWxat2WSXWDCnDYEFvM7Q
embedding
embedding/embeddings_0
fit/20210102-174327/train
lstm
lstm/lstm_cell/bias_0
lstm/lstm_cell/kernel_0
lstm/lstm_cell/recurrent_kernel_0


dropout
num_relu_units
train.epoch_accuracy
train.epoch_loss

CODING

CODING SNIPPETS

Simpsons Character Recognition using DCNN (.ipynb Google Colab) –

The training phase of the DCNN model in Google Colab.

In [ ]:
```
from google.colab import files
uploaded = files.upload()
```



```
Saving kaggle.json to kaggle.json
```

In [ ]:
```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!echo '{"username":"kushgabani","key":"34517a5e1368a343ff4b14f0e624f5fd"}' > /root/.kaggle/kaggle.json
!kaggle datasets download -d alexattia/the-simpsons-characters-dataset
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
Downloading the-simpsons-characters-dataset.zip to /content
 99% 1.07G/1.08G [00:14<00:00, 69.1MB/s]
100% 1.08G/1.08G [00:14<00:00, 82.0MB/s]
```

In [ ]:
```
!unzip /content/the-simpsons-characters-dataset.zip -d DataSet
```

```
Streaming output truncated to the last 5000 lines.
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1270.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1271.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1272.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1273.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1274.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1275.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1276.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1277.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1278.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1279.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1280.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1281.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1282.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1283.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1284.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1285.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1286.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1287.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1288.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1289.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1290.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1291.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1292.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1293.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1294.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1295.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1296.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1297.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1298.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1299.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1300.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1301.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1302.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1303.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1304.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1305.jpg
  inflating: DataSet/simpsons_dataset/simpsons_dataset/moe_szyslak/pic_1306.jpg
```

```
  inflating: DataSet/simpsons_dataset/waylon_smithers/pic_0174.jpg
  inflating: DataSet/simpsons_dataset/waylon_smithers/pic_0175.jpg
  inflating: DataSet/simpsons_dataset/waylon_smithers/pic_0176.jpg
  inflating: DataSet/simpsons_dataset/waylon_smithers/pic_0177.jpg
  inflating: DataSet/simpsons_dataset/waylon_smithers/pic_0178.jpg
  inflating: DataSet/simpsons_dataset/waylon_smithers/pic_0179.jpg
  inflating: DataSet/simpsons_dataset/waylon_smithers/pic_0180.jpg
  inflating: DataSet/weights.best.hdf5
```

In [ ]:
```
%tensorflow_version 2.x
import os
import shutil
import warnings
import numpy as np
from numpy.random import seed
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
import cv2
from imblearn.over_sampling import RandomOverSampler
import tensorflow as tf
from tensorflow import keras
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense, Dropout
from keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.models import load_model
from sklearn.model_selection import GridSearchCV
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.metrics import confusion_matrix
from keras.utils.vis_utils import plot_model

warnings.filterwarnings("ignore")
```

In [ ]:
```
from tensorboard.plugins.hparams import api as hp
import datetime

!rm -rf logs/image

log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(
    log_dir=log_dir, histogram_freq=1)
hparams_callback = hp.KerasCallback(log_dir, {
    'num_relu_units': 512,
    'dropout': 0.2
})
```

In [ ]:
```
training_path = "./DataSet/simpsons_dataset/"
testing_path = "./DataSet/kaggle_simpson_testset/"
CHARACTERS = os.listdir(training_path)
CHARACTERS.remove("simpsons_dataset")
shutil.rmtree(training_path + "simpsons_dataset")
CHARACTERS = sorted(CHARACTERS)
CHARACTERS
```

Out[ ]:
```
['abraham_grampa_simpson',
 'agnes_skinner',
 'apu_nahasapeemapetilon',
 'barney_gumble',
 'bart_simpson',
 'carl_carlson',
 'charles_montgomery_burns',
 'chief_wiggum',
 'cletus_spuckler',
 'comic_book_guy',
 'disco_stu',
 'edna_krabappel',
 'fat_tony',
 'gil',
 'groundskeeper_willie',
 'homer_simpson',
 'kent_brockman',
 'krusty_the_clown',
 'lenny_leonard',
 'lionel_hutz',
 'lisa_simpson',
 'maggie_simpson',
 'marge_simpson',
 'martin_prince',
 'mayor_quimby',
 'milhouse_van_houten',
 'miss_hoover',
 'moe_szyslak',
 'ned_flanders',
 'nelson_muntz',
 'otto_mann',
 'patty_bouvier',
 'principal_skinner',
 'professor_john_frink',
 'rainier_wolfcastle',
 'ralph_wiggum',
 'selma_bouvier',
 'sideshow_bob',
 'sideshow_mel',
 'snake_jailbird',
 'troy_mcclure',
 'waylon_smithers']
```

In [ ]:
```
os.remove("./DataSet/annotation.txt")
os.remove("./DataSet/characters_illustration.png")
os.remove("./DataSet/number_pic_char.csv")
os.remove("./DataSet/weights.best.hdf5")
```

In [ ]:
```
character_count = {}
to_be_removed = []
print("Characters removed from the set because of Under-Sampling")
for char in CHARACTERS:
  for r, _, files in os.walk(training_path + char):
    samples = len(files)
    # if samples < 200:
    #   print(char + " : " + str(samples))
    #   to_be_removed.append(char)
    #   CHARACTERS.remove(char)
    # else:
    character_count[char] = samples

CHARACTERS
```

```
Characters removed from the set because of Under-Sampling
```

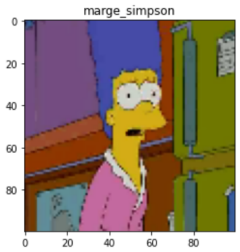
Out[ ]:
```
['abraham_grampa_simpson',
 'agnes_skinner',
 'apu_nahasapeemapetilon',
 'barney_gumble',
 'bart_simpson',
 'carl_carlson',
 'charles_montgomery_burns',
 'chief_wiggum',
 'cletus_spuckler',
 'comic_book_guy',
 'disco_stu',
 'edna_krabappel',
 'fat_tony',
 'gil',
 'groundskeeper_willie',
 'homer_simpson',
 'kent_brockman',
 'krusty_the_clown',
 'lenny_leonard',
 'lionel_hutz',
 'lisa_simpson',
 'maggie_simpson',
 'marge_simpson',
 'martin_prince',
 'mayor_quimby',
 'milhouse_van_houten',
 'miss_hoover',
 'moe_szyslak',
 'ned_flanders',
 'nelson_muntz',
 'otto_mann',
 'patty_bouvier',
 'principal_skinner',
 'professor_john_frink',
 'rainier_wolfcastle',
 'ralph_wiggum',
 'selma_bouvier',
 'sideshow_bob',
 'sideshow_mel',
 'snake_jailbird',
 'troy_mcclure',
 'waylon_smithers']
```

In [ ]:
```
os.rename(training_path, "./DataSet/train/")
os.rename(testing_path, "./DataSet/test/")

training_path = "./DataSet/train/"
testing_path = "./DataSet/test/"

shutil.move(testing_path + "kaggle_simpson_testset", "./DataSet")
shutil.rmtree(testing_path)
os.rename("./DataSet/kaggle_simpson_testset", testing_path)

shutil.rmtree("./sample_data")
```

In [ ]:
```
character_map = dict(zip(np.arange(len(CHARACTERS)), CHARACTERS))

X_train = []
y_train = []

X_test = []
y_labels = []

CHARACTER_COUNT = len(CHARACTERS)

IMG_WIDTH = IMG_HEIGHT = 100

for num_map, character in character_map.items():
  for r, _, files in os.walk(training_path + character):
    for image in files:
      img = cv2.imread(training_path + character + "/" + image)
      img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
      img = cv2.resize(img, (IMG_WIDTH, IMG_HEIGHT))
      X_train.append(img)
      y_train.append(num_map)

X_train = np.array(X_train)
y_train = to_categorical(np.array(y_train), len(np.unique(y_train)))

for r, _, files in os.walk(testing_path):
  for image in files:
      img = cv2.imread(testing_path + image)
      img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
      img = cv2.resize(img, (IMG_WIDTH, IMG_HEIGHT))
      X_test.append(img)
      filename = '_'.join(image.split("_")[:-1])
      y_labels.append(filename)

X_test = np.array(X_test)
y_test = np.zeros((len(X_test), y_train.shape[1]))

for i, label in enumerate(y_labels):
  lbl_index = list(character_map.values()).index(label)
  y_test[i][lbl_index] = 1
```

In [ ]:
```
plt.imshow(X_train[12340])
plt.title(character_map[list(y_train[12340]).index(1)])
print((y_train.shape, y_test.shape, len(CHARACTERS)))
```

```
((20933, 42), (990, 42), 42)
```



In [ ]:
```
%load_ext tensorboard
from datetime import datetime
import io
import itertools
from six.moves import range
import sklearn.metrics

!rm -rf logs/image
logdir = "logs/train_data/" + datetime.now().strftime("%Y%m%d-%H%M%S")
# Creates a file writer for the log directory.
file_writer = tf.summary.create_file_writer(logdir)

with file_writer.as_default():
  tf.summary.image("5 training data examples", X_train[500:505], max_outputs=25, step=0)

%tensorboard --logdir logs/train_data
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Reusing TensorBoard on port 6006 (pid 726), started 0:01:01 ago. (Use '!kill 726' to kill it.)
<IPython.core.display.Javascript object>
```

In [ ]:
```
print("X_train's shape before : " + str(X_train.shape))
print("y_train's shape before : " + str(y_train.shape))

ros = RandomOverSampler(sampling_strategy = "minority", random_state = 42)
X_train, y_train = ros.fit_sample(X_train.reshape(len(X_train), IMG_WIDTH * IMG_HEIGHT * 3), y_train)

X_train = X_train.reshape(-1, 100, 100, 3)

print("X_train's shape after : " + str(X_train.shape))
print("y_train's shape after : " + str(y_train.shape))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/externals/six.py:31: FutureWarning: The module is deprecated in version 0.21 and will be r
emoved in version 0.23 since we've dropped support for Python 2.7. Please rely on the official version of six (https://pypi.org/project/s
ix/).
  "(https://pypi.org/project/six/).", FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: The sklearn.neighbors.base module is  deprecated
in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.neighb
ors. Anything that cannot be imported from sklearn.neighbors is now part of the private API.
  warnings.warn(message, FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function safe_indexing is deprecated; safe_indexin
g is deprecated in version 0.22 and will be removed in version 0.24.
  warnings.warn(msg, category=FutureWarning)
X_train's shape before : (20933, 100, 100, 3)
y_train's shape before : (20933, 42)
X_train's shape after : (23176, 100, 100, 3)
y_train's shape after : (23176, 42)
```

In [ ]:
```
modified_character_counts = dict(zip(list(CHARACTERS), [0 for char in CHARACTERS]))
for y in y_train:
  character = CHARACTERS[np.argmax(y)]
  modified_character_counts[character] += 1
modified_character_counts
```

Out[ ]:
```
{'abraham_grampa_simpson': 913,
 'agnes_skinner': 42,
 'apu_nahasapeemapetilon': 623,
 'barney_gumble': 106,
 'bart_simpson': 1342,
 'carl_carlson': 98,
 'charles_montgomery_burns': 1193,
 'chief_wiggum': 986,
 'cletus_spuckler': 47,
 'comic_book_guy': 469,
 'disco_stu': 8,
 'edna_krabappel': 457,
 'fat_tony': 27,
 'gil': 27,
 'groundskeeper_willie': 121,
 'homer_simpson': 2246,
 'kent_brockman': 498,
 'krusty_the_clown': 1206,
 'lenny_leonard': 310,
 'lionel_hutz': 2246,
 'lisa_simpson': 1354,
 'maggie_simpson': 128,
 'marge_simpson': 1291,
 'martin_prince': 71,
```

In [ ]:
```
%%time
np.random.seed(42)

num_classes = len(CHARACTERS)
model = Sequential([
                    Conv2D(32, (3, 3), activation = "relu", padding = "same", input_shape = (IMG_WIDTH, IMG_HEIGHT, 3)),
                    MaxPool2D((2, 2)),
                    Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                    Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                    MaxPool2D((2, 2)),
                    Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                    MaxPool2D((2, 2)),
                    Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                    Conv2D(128, (3, 3), padding = "same", activation = "relu"),
                    MaxPool2D((2, 2)),
                    Conv2D(128, (3, 3), padding = "same", activation = "relu"),
                    Flatten(),
                    Dense(70, activation = "relu"),
                    Dropout(0.3),
                    Dense(120, activation = "relu"),
                    Dropout(0.3),
                    Dense(num_classes, activation = "softmax")
                    ])

model.compile(optimizer = "adam",
              loss = "categorical_crossentropy",
              metrics = ["accuracy"])

batch_size = 32
history = model.fit(X_train, y_train,
                    epochs = 20,
                    batch_size = batch_size,
                    validation_data = (X_test, y_test))
```

In [ ]:
```
%%time
# Hyperparameter tuning (optimizer)

def create_model(optimizer = "adam"):
  num_classes = len(CHARACTERS)
  model = Sequential([
                      Conv2D(32, (3, 3), activation = "relu", padding = "same", input_shape = (IMG_WIDTH, IMG_HEIGHT, 3)),
                      MaxPool2D((2, 2)),
                      Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                      Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                      MaxPool2D((2, 2)),
                      Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                      MaxPool2D((2, 2)),
                      Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                      Conv2D(128, (3, 3), padding = "same", activation = "relu"),
                      MaxPool2D((2, 2)),
                      Conv2D(128, (3, 3), padding = "same", activation = "relu"),
                      Flatten(),
                      Dense(70, activation = "relu"),
                      Dropout(0.3),
                      Dense(120, activation = "relu"),
                      Dropout(0.3),
                      Dense(num_classes, activation = "softmax")
                      ])

  model.compile(optimizer = optimizer,
                loss = "categorical_crossentropy",
                metrics = ["accuracy"])
  return model


model = KerasClassifier(build_fn = create_model, epochs = 30, batch_size = 32)
param_dist = {
               "optimizer" : ["sgd", "adam", "rmsprop"]
             }

gridCV = GridSearchCV(estimator = model,
                      param_grid = param_dist,
                      n_jobs = -1,
                      cv = 3)

grid_result = gridCV.fit(X_train, y_train)

print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

# Adam came out to be the best optimizer out of Adam, Rmsprop, SGD
```

```
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:691: UserWarning: A worker stopped while some jobs were given to the executor. This can be caused by a too short worker timeout or by a memory leak.
  "timeout or by a memory leak.", UserWarning
```

In [ ]:
```
%%time
# Hyperparameter tuning (batch_size, epochs)
from sklearn.model_selection import RandomizedSearchCV
from keras.wrappers.scikit_learn import KerasClassifier

# del model

def create_model():
  num_classes = len(CHARACTERS)
  model = Sequential([
                      Conv2D(32, (3, 3), activation = "relu", padding = "same", input_shape = (IMG_WIDTH, IMG_HEIGHT, 3)),
                      MaxPool2D((2, 2)),
                      Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                      Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                      MaxPool2D((2, 2)),
                      Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                      MaxPool2D((2, 2)),
                      Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                      Conv2D(128, (3, 3), padding = "same", activation = "relu"),
                      MaxPool2D((2, 2)),
                      Conv2D(128, (3, 3), padding = "same", activation = "relu"),
                      Flatten(),
                      Dense(70, activation = "relu"),
                      Dropout(0.3),
                      Dense(120, activation = "relu"),
                      Dropout(0.3),
                      Dense(num_classes, activation = "softmax")
                      ])

  model.compile(optimizer = "adam",
                loss = "categorical_crossentropy",
                metrics = ["accuracy"])
  return model
seed = 42

model = KerasClassifier(build_fn = create_model, verbose = 1)
param_dist = {
                "batch_size" : [30, 50, 75],
                "epochs" : [20, 25, 30]
              }

randomCV = RandomizedSearchCV(estimator = model,
                              param_distributions = param_dist,
                              n_iter = 4,
                              n_jobs = -1,
                              cv = 2)

randomCV_result = randomCV.fit(X_train, y_train)

print("Best: %f using %s" % (randomCV_result.best_score_, randomCV_result.best_params_))
means = randomCV_result.cv_results_['mean_test_score']
stds = randomCV_result.cv_results_['std_test_score']
params = randomCV_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

In [ ]:
```
%%time
# Hyperparameter tuning (batch_size, epochs)
np.random.seed(42)
# del model

def create_model(dropout_rate = 0):
  num_classes = len(CHARACTERS)
  model = Sequential([
                      Conv2D(32, (3, 3), activation = "relu", padding = "same", input_shape = (IMG_WIDTH, IMG_HEIGHT, 3)),
                      MaxPool2D((2, 2)),
                      Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                      Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                      MaxPool2D((2, 2)),
                      Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                      MaxPool2D((2, 2)),
                      Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                      Conv2D(128, (3, 3), padding = "same", activation = "relu"),
                      MaxPool2D((2, 2)),
                      Conv2D(128, (3, 3), padding = "same", activation = "relu"),
                      Flatten(),
                      Dense(70, activation = "relu"),
                      Dropout(0.3),
                      Dense(120, activation = "relu"),
                      Dropout(dropout_rate),
                      Dense(num_classes, activation = "softmax")
                      ])

  model.compile(optimizer = "adam",
                loss = "categorical_crossentropy",
                metrics = ["accuracy"])
  return model

model = KerasClassifier(build_fn = create_model, verbose = 1, epochs = 30, batch_size = 50)
param_dist = {
                "dropout_rate" : [0, 0.2, 0.4]
              }

gridCV = GridSearchCV(estimator = model,
                              param_grid = param_dist,
                              n_jobs = -1,
                              cv = 2)

gridcv_result = gridCV.fit(X_train, y_train)

print("Best: %f using %s" % (gridcv_result.best_score_, gridcv_result.best_params_))
means = gridcv_result.cv_results_['mean_test_score']
stds = gridcv_result.cv_results_['std_test_score']
params = gridcv_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

In [ ]:
```
%%time
# del model
num_classes = len(CHARACTERS)
model = Sequential([
                    Conv2D(32, (3, 3), activation = "relu", padding = "same", input_shape = (IMG_WIDTH, IMG_HEIGHT, 3)),
                    MaxPool2D((2, 2)),
                    Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                    Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                    MaxPool2D((2, 2)),
                    Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                    MaxPool2D((2, 2)),
                    Conv2D(64, (3, 3), padding = "same", activation = "relu"),
                    Conv2D(128, (3, 3), padding = "same", activation = "relu"),
                    MaxPool2D((2, 2)),
                    Conv2D(128, (3, 3), padding = "same", activation = "relu"),
                    Flatten(),
                    Dense(70, activation = "relu"),
                    Dropout(0.3),
                    Dense(120, activation = "relu"),
                    Dropout(0.3),
                    Dense(num_classes, activation = "softmax")
                    ])

model.compile(optimizer = "adam",
              loss = "categorical_crossentropy",
              metrics = ["accuracy"])

reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                              factor=0.2,
                              patience=3,
                              min_lr=0.0001,
                              verbose = 1,
                              mode = "min")

batch_size = 50
history = model.fit(X_train, y_train,
                    epochs = 30,
                    batch_size = batch_size,
                    validation_data = (X_test, y_test),
                    callbacks = [reduce_lr, tensorboard_callback, hparams_callback])
```

```
CPU times: user 99.5 ms, sys: 0 ns, total: 99.5 ms
Wall time: 99.5 ms
```

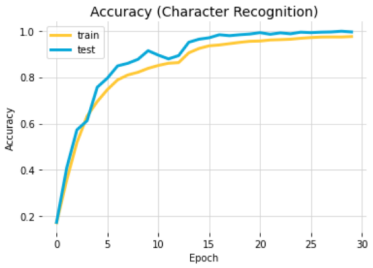
In [ ]:
```
!tensorboard dev upload --logdir ./logs \
  --name "Simpsons Character Recognition using Deep Convolutional Neural Networks  " \
  --description "Training results from https://colab.research.google.com/drive/11KYv4B26DVD5YIzhzuvtZV3NnEcR0pmN#scrollTo=kHe3koM4DeOY" \
  --one_shot
```

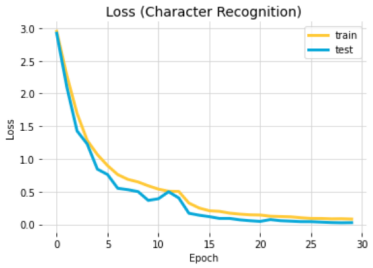
```
2021-01-02 17:05:45.528386: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudar
t.so.10.1

***** TensorBoard Uploader *****

This will upload your TensorBoard logs to https://tensorboard.dev/ from
the following directory:

./logs

This TensorBoard will be visible to everyone. Do not upload sensitive
data.

Your use of this service is subject to Google's Terms of Service
<https://policies.google.com/terms> and Privacy Policy
<https://policies.google.com/privacy>, and TensorBoard.dev's Terms of Service
<https://tensorboard.dev/policy/terms/>.

This notice will not be shown again while you are logged into the uploader.
To log out, run `tensorboard dev auth revoke`.

Continue? (yes/NO) yes

Please visit this URL to authorize this application: https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=373649185512-
8v619h5kft38l4456nm2dj4ubeqsrvh6.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=openid+https%3A%2F%2Fw
ww.googleapis.com%2Fauth%2Fuserinfo.email&state=o0bNafYjJeBVCy4eCcbUX30x5UP7UQ&prompt=consent&access_type=offline
Enter the authorization code: 4/1AY0e-g5YvGyiJ-Gy3jZM7BMIwAzD3Pu3wWt0OrnxhsCj80eDqo_iamalv4w


New experiment created. View your TensorBoard at: https://tensorboard.dev/experiment/GyTX0fJIQzqpm4IVc2NQ0Q/

[2021-01-02T17:06:28] Started scanning logdir.
[2021-01-02T17:06:30] Total uploaded: 120 scalars, 603 tensors (938.9 kB), 1 binary objects (80.0 kB)
[2021-01-02T17:06:30] Done scanning logdir.


Done. View your TensorBoard at https://tensorboard.dev/experiment/GyTX0fJIQzqpm4IVc2NQ0Q/
```

In [ ]:
```
test_loss, test_acc = model.evaluate(X_test, y_test)
print("Test Loss : " + str(test_loss))
print("Test Accuracy : " + str(test_acc))
```

```
31/31 [==============================] - 0s 9ms/step - loss: 0.0261 - accuracy: 0.9949
Test Loss : 0.026103010401129723
Test Accuracy : 0.9949495196342468
```

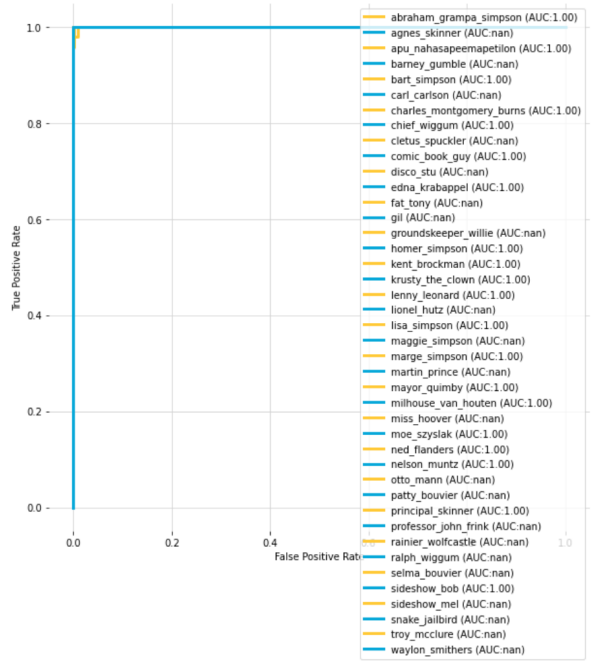
```
colors = cycler('color',
                ['#ffc936', '#07a8d9'])
plt.rc('axes', facecolor='#FFFFFF', edgecolor='none',
       axisbelow=True, grid=True, prop_cycle = colors)
plt.rc('grid', color='lightgray', linestyle='solid')
plt.rc('patch', edgecolor='#fafafa')
plt.rc('lines', linewidth=3)
plt.plot(history.history["accuracy"])
plt.plot(history.history["val_accuracy"])

plt.title("Accuracy (Character Recognition)", fontsize = "14")
plt.xlabel('Epoch')
plt.ylabel("Accuracy")
plt.legend(['train', 'test'])
plt.show()

plt.plot(history.history["loss"])
plt.plot(history.history["val_loss"])
plt.title("Loss (Character Recognition)", fontsize = "14")
plt.xlabel('Epoch')
plt.ylabel("Loss")
plt.legend(['train', 'test'])
plt.show();
```





In [ ]:
```
from sklearn.metrics import roc_curve, auc
import warnings
warnings.filterwarnings("ignore")

y_preds = model.predict(X_test)

fig, ax = plt.subplots(figsize = (10, 10))
for i, val in enumerate(CHARACTERS):
    fpr, tpr, thresholds = roc_curve(y_test[:,i].astype(int), y_preds[:,i])
    ax.plot(fpr, tpr, label = '%s (AUC:%0.2f)'  % (val, auc(fpr, tpr)))
ax.legend()
ax.set_xlabel('False Positive Rate')
ax.set_ylabel('True Positive Rate')
```

Out[ ]:
```
Text(0, 0.5, 'True Positive Rate')
```



Script Generation using LSTMS (.ipynb Google Colab) –

Training phase of the Sequence-to-Vector LSTM model for script generation.

In [ ]:
```
%tensorflow_version 2.x
import io
import os
import sys
import warnings
from datetime import datetime
import shutil
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cycler
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Embedding
from keras.optimizers import Adam, RMSprop, SGD
from keras.callbacks import ModelCheckpoint
from keras.utils.vis_utils import plot_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorboard.plugins.hparams import api as hp

try:
  shutil.rmtree("./sample_data")
except:
  pass
```

In [ ]:
```
!rm -rf logs/image

log_dir="logs/fit/" + datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(
    log_dir=log_dir, histogram_freq=1)
hparams_callback = hp.KerasCallback(log_dir, {
    'num_relu_units': 512,
    'dropout': 0.2
})
```

In [ ]:
```
with io.open("./drive/MyDrive/moes_tavern_lines.txt", encoding = "utf-8-sig") as f:
  text = f.read().lower()

print("Sample training data :\n")
text = text[81:]
print(text[:500])
```

```
Sample training data :

moe_szyslak: (into phone) moe's tavern. where the elite meet to drink.
bart_simpson: eh, yeah, hello, is mike there? last name, rotch.
moe_szyslak: (into phone) hold on, i'll check. (to barflies) mike rotch. mike rotch. hey, has anybody seen mike rotch, lately?
moe_szyslak: (into phone) listen you little puke. one of these days i'm gonna catch you, and i'm gonna carve my name on your back with an
ice pick.
moe_szyslak: what's the matter homer? you're not your normal effervescent self.
homer_simp
```

In [ ]:
```
data = text.split("\n")
print(data[0])
data = " ".join(data)
print(data[:100])
```

```
moe_szyslak: (into phone) moe's tavern. where the elite meet to drink.
moe_szyslak: (into phone) moe's tavern. where the elite meet to drink. bart_simpson: eh, yeah, hello
```

In [ ]:
```
punctuations = {
    '.' : "period",
    '[' : "leftbraces",
    ']' : "rightbraces",
    '(' : "leftparen",
    ')' : "rightparen",
    ';' : "semicolon",
    '$' : "price",
    '%' : "percentage",
    '&' : "and",
    '#' : "hash",
    '\n' : "newline",
    ':' : "colon",
    "'" : "apostrophe",
    '/' : "or",
    '"' : "quote",
    ',' : "comma",
    '?' : "question",
    '*' : "asterisk",
    '!' : "exclamation",
    '-' : "hyphen",
}

def preprocess_text(text):
  for punc, alt in punctuations.items():
    text = text.replace(punc, ' ' + alt + ' ')
  text = text.replace("\n", punctuations["\n"])
  tokens = text.split()
  return tokens

tokens = preprocess_text(text)
print(tokens[:101])
```

```
['moe_szyslak', 'colon', 'leftparen', 'into', 'phone', 'rightparen', 'moe', 'apostrophe', 's', 'tavern', 'period', 'where', 'the', 'elit
e', 'meet', 'to', 'drink', 'period', 'newline', 'bart_simpson', 'colon', 'eh', 'comma', 'yeah', 'comma', 'hello', 'comma', 'is', 'mike',
'there', 'question', 'last', 'name', 'comma', 'rotch', 'period', 'newline', 'moe_szyslak', 'colon', 'leftparen', 'into', 'phone', 'rightp
aren', 'hold', 'on', 'comma', 'i', 'apostrophe', 'll', 'check', 'period', 'leftparen', 'to', 'barflies', 'rightparen', 'mike', 'rotch',
'period', 'mike', 'rotch', 'period', 'hey', 'comma', 'has', 'anybody', 'seen', 'mike', 'rotch', 'comma', 'lately', 'question', 'newline',
'moe_szyslak', 'colon', 'leftparen', 'into', 'phone', 'rightparen', 'listen', 'you', 'little', 'puke', 'period', 'one', 'of', 'these', 'd
ays', 'i', 'apostrophe', 'm', 'gonna', 'catch', 'you', 'comma', 'and', 'i', 'apostrophe', 'm', 'gonna', 'carve', 'my']
```

In [ ]:
```
print("length of tokens : " + str(len(tokens)))
print("number of unique tokens : " + str(len(set(tokens))))
```

```
length of tokens : 79756
number of unique tokens : 6355
```

In [ ]:
```
sequence_length = 50 + 1
lines = []

for i in range(sequence_length, len(tokens)):
  sequence = tokens[i - sequence_length : i]
  lines.append(' '.join(sequence))

print(len(lines))
```

```
79705
```

In [ ]:
```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(lines)
sequences = tokenizer.texts_to_sequences(lines)
sequences = pad_sequences(sequences, maxlen = sequence_length, truncating='pre')
len(sequences[0])
```

Out[ ]:
```
51
```

In [ ]:
```
sequences = np.array(sequences)
x, y = sequences[:, :-1], sequences[:, -1]
sequence_length = x.shape[1]
sequence_length
```

Out[ ]:
```
50
```

In [ ]:
```
vocab_size = len(tokenizer.word_index) + 1
y = to_categorical(y, num_classes=vocab_size)
vocab_size
```

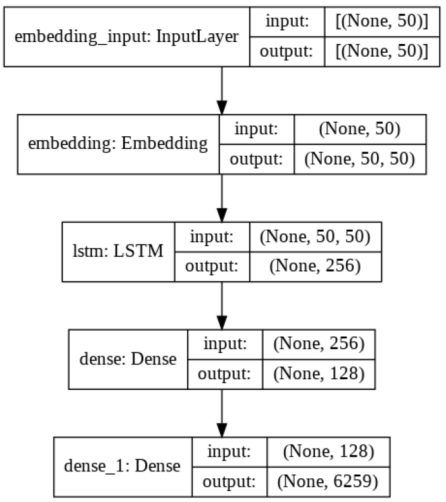
Out[ ]:
```
6259
```

In [ ]:
```
model = Sequential()
model.add(Embedding(vocab_size, 50, input_length=x.shape[1]))
model.add(LSTM(256))
model.add(Dense(128, activation='relu'))
model.add(Dense(vocab_size, activation='softmax'))

model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 50, 50)            312950
_________________________________________________________________
lstm (LSTM)                  (None, 256)               314368
_________________________________________________________________
dense (Dense)                (None, 128)               32896
_________________________________________________________________
dense_1 (Dense)              (None, 6259)              807411
=================================================================
Total params: 1,467,625
Trainable params: 1,467,625
Non-trainable params: 0
```

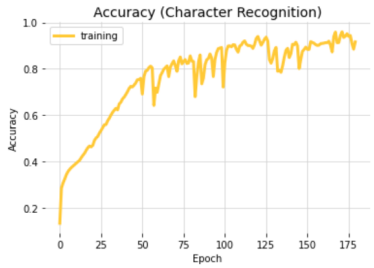
In [ ]:
```
plot_model(model, to_file='script-generation-model-architecture.png', show_shapes=True, show_layer_names=True)
```

Out[ ]:



In [ ]:
```
history = model.fit(x, y, epochs = 180, batch_size = 128, callbacks=[tensorboard_callback, hparams_callback])
```

In [ ]:
```
colors = cycler('color',
                ['#ffc936', '#07a8d9'])
plt.rc('axes', facecolor='#FFFFFF', edgecolor='none',
       axisbelow=True, grid=True, prop_cycle = colors)
plt.rc('grid', color='lightgray', linestyle='solid')
plt.rc('patch', edgecolor='#fafafa')
plt.rc('lines', linewidth=3)

plt.plot(history.history["accuracy"])
plt.title("Accuracy (Character Recognition)", fontsize = "14")
plt.xlabel('Epoch')
plt.ylabel("Accuracy")
plt.legend(['training'])
plt.show()

plt.plot(history.history["loss"])
plt.title("Loss (Character Recognition)", fontsize = "14")
plt.xlabel('Epoch')
plt.ylabel("Loss")
plt.legend(['training'])
plt.show();
```



In [ ]:
```
def generate_text_seq(model, tokenizer, text_seq_length, seed_text, n_words):
  sys.stdout.write(seed_text + "\n");
  text = []
  for _ in range(n_words):
    encoded = tokenizer.texts_to_sequences([seed_text])[0]
    encoded = pad_sequences([encoded], maxlen = text_seq_length, truncating='pre')
    y_predict = model.predict_classes(encoded)

    predicted_word = ''
    for word, index in tokenizer.word_index.items():
      if index == y_predict:
        predicted_word = word
        break
    seed_text = seed_text + ' ' + predicted_word

    if predicted_word in list(punctuations.values()):
      predicted_word = list(punctuations.keys())[list(punctuations.values()).index(predicted_word)]

    text.append(predicted_word)
  return ' '.join(text)
```

In [ ]:
```
warnings.filterwarnings("ignore")
test_text = "marge well leave it to good old Mary Bailey to finally step in and do something about that hideous genetic mutation homer ma
ry bailey well if i was governor i d sure find better things to do with my time marge like what homer like getting"
print(generate_text_seq(saved_model, tokenizer, sequence_length, test_text, 100))
# print(generate_text_seq(saved_model, tokenizer, sequence_length, " ".join(tokens[120:170]), 100))
```

```
marge well leave it to good old Mary Bailey to finally step in and do something about that hideous genetic mutation homer mary bailey wel
l if i was governor i d sure find better things to do with my time marge like what homer like getting
disappointed to say by my mother , buy ran his alley .
 lenny leonard : keep everyone here real optimistic literature celebrities ? ( cute ) oh , as it ' s so strong at us out from now . i don
' t think it ' s given me a beer .


 larry : oh , duff lovers ! does anyone in the only dangerous brunch spectacular .


 smitty : you know that , ain ' t that smile .


 mrs . powers : ( suddenly nervous ) you
```

Requirements.txt –

Each and every dependency used to create the web interface and deploy the active models for testing.

```
absl-py==0.11.0
astunparse==1.6.3
cachetools==4.2.0
certifi==2020.12.5
cffi==1.14.4
chardet==4.0.0
click==7.1.2
Flask==1.1.2
flatbuffers==1.12
gast==0.3.3
gevent==20.9.0
google-auth==1.24.0
google-auth-oauthlib==0.4.2
google-pasta==0.2.0
greenlet==0.4.17
grpcio==1.32.0
gunicorn==20.0.4
h5py==2.10.0
idna==2.10
itsdangerous==1.1.0
Jinja2==2.11.2
Keras-Preprocessing==1.1.2
Markdown==3.3.3
MarkupSafe==1.1.1
numpy==1.19.3
oauthlib==3.1.0
opencv-python==4.4.0.46
opt-einsum==3.3.0
protobuf==3.14.0
pyasn1==0.4.8
pyasn1-modules==0.2.8
pycparser==2.20
requests==2.25.1
requests-oauthlib==1.3.0
rsa==4.6
six==1.15.0
tensorboard==2.4.0
tensorboard-plugin-wit==1.7.0
tensorflow-cpu==2.4.0
tensorflow-estimator==2.4.0
termcolor==1.1.0
typing-extensions==3.7.4.3
urllib3==1.26.2
Werkzeug==1.0.1
wrapt==1.12.1
zope.event==4.5.0
zope.interface==5.2.0
```

App.py-

Flask back-end with Routes for the website and API for to get predictions from the train model for both the modules.

```
import os
import io
import sys
import warnings
import shutil
import numpy as np
import cv2
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from flask import Flask, url_for, request, render_template
from werkzeug.utils import secure_filename

warnings.filterwarnings("ignore")

app = Flask(__name__)

characters = ['Abraham Grampa Simpson',
 'Agnes Skinner',
 'Apu Nahasapeemapetilon',
 'Barney Gumble',
 'Bart Simpson',
 'Carl Carlson',
 'Charles Montgomery Burns',
 'Chief Wiggum',
 'Cletus Spuckler',
 'Comic Book Guy',
 'Disco Stu',
 'Edna Krabappel',
 'Fat Tony',
 'Gil',
 'Groundskeeper Willie',
 'Homer Simpson',
 'Kent Brockman',
 'Krusty The Clown',
 'Lenny Leonard',
 'Lionel Hutz',
 'Lisa Simpson',
 'Maggie Simpson',
 'Marge Simpson',
```

```
 'Martin Prince',
 'Mayor Quimby',
 'Milhouse Van Houten',
 'Miss Hoover',
 'Moe Szyslak',
 'Ned Flanders',
 'Nelson Muntz',
 'Otto Mann',
 'Patty Bouvier',
 'Principal Skinner',
 'Professor John Frink',
 'Rainier Wolfcastle',
 'Ralph Wiggum',
 'Selma Bouvier',
 'Sideshow Bob',
 'Sideshow Mel',
 'Snake Jailbird',
 'Troy Mcclure',
 'Waylon Smithers']

punctuations = {
    '.' : "period",
    '[' : "leftbraces",
    ']' : "rightbraces",
    '(' : "leftparen",
    ')' : "rightparen",
    ';' : "semicolon",
    '$' : "price",
    '%' : "percentage",
    '&' : "and",
    '#' : "hash",
    '\n' : "newline",
    ':' : "colon",
    "'" : "apostrophe",
    '/' : "or",
    '"' : "quote",
    ',' : "comma",
    '?' : "question",
    '*' : "asterisk",
    '!' : "exclamation",
    '-' : "hyphen",
}
```

```python
def get_prediction_from_cnn(file_path):
    model = load_model("./neural_networks/character-recognition-model.h5")

    image = cv2.imread(file_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image, (100, 100))
    image = np.reshape(image, (1, 100, 100, 3))
    prediction = model.predict(image)
    char_index = np.argmax(prediction, axis = 1)
    return characters[char_index[0]]

def preprocess_text(text):
  for punc, alt in punctuations.items():
    text = text.replace(punc, ' ' + alt + ' ')
  text = text.replace("\n", punctuations["\n"])
  tokens = text.split()
  return tokens

def get_tokenizer():
    with io.open("./script_data/moes_tavern_lines.txt", encoding = "utf-8-sig") as f:
        text = f.read().lower()
        text = text[81:]

    print("Loaded script as a text file.")

    tokens = preprocess_text(text)
    print("Preprocessed data.")
    sequence_length = 50 + 1
    lines = []

    for i in range(sequence_length, len(tokens)):
        sequence = tokens[i - sequence_length : i]
        lines.append(' '.join(sequence))
    print("Built sequences")

    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(lines)
    print("Fit tokenizer on data.")
    return tokenizer
```

```python
def generate_script_from_lstm(text_input, n_words):
    sequence_length = 50
    tokenizer = get_tokenizer()
    vocab_size = len(tokenizer.word_index) + 1

    model = load_model("./neural_networks/script-generation-model-with-punctuation.h5")
    print("Loaded Model")

    text = ""
    for _ in range(n_words):
        encoded = tokenizer.texts_to_sequences([text_input])[0]
        encoded = pad_sequences([encoded], maxlen = sequence_length, truncating='pre')

        y_predict = model.predict_classes(encoded)

        predicted_word = ''
        for word, index in tokenizer.word_index.items():
            if index == y_predict:
                predicted_word = word
                break
        text_input = text_input + ' ' + predicted_word

        if predicted_word in list(punctuations.values()):
            predicted_word = list(punctuations.keys())[list(punctuations.values()).index(predicted_word)]
        text += predicted_word + " "
        if predicted_word == ".":
            text += "<br/>"
    return text

@app.route('/', methods=['GET'])
def home():
    return render_template('home.html')

@app.route('/character-recognition', methods=['GET'])
def index():
    return render_template('character-recognition.html')
```

```python

@app.route('/script-generation', methods=['GET', 'POST'])
def script_generation():
    if request.method == "GET":
        return render_template('script-generation.html', result = False)
    else:
        beginning = request.form['input']
        input_text = beginning.replace("<br/>", "")
        n_words = int(request.form['word']) if request.form['word'] else 100
        output = generate_script_from_lstm(input_text, n_words)
        output = beginning + "<br/>" + output
        return render_template('script-generation.html', result = output)

@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == "POST":
        print("Reached Here")
        f = request.files['file']
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)

        prediction = get_prediction_from_cnn(file_path)
        return prediction
    return 'GET Request /predict'

if __name__ == "__main__":
    app.run(debug=True)
```

Base.html -

A skeleton for the web pages that will use this format.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    {% block title %}{% endblock %}
    {% block css %}{% endblock %}

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css" i
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/2.6.0/umd/popper.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

  </head>
  <body>
      {% block content  %}{% endblock %}
  </body>

  {% block scriptFooter %}{% endblock %}
</html>
```

Home.html -

The home page of the web application from where you can access both the modules – Character Recognition and Script Generation.

```
{% extends "base.html" %}


{% block title %}
  <title>Simpsons - Home</title>
{% endblock %}

{% block css %}
    <link rel="stylesheet" href="{{ url_for('static', filename='css/home.css') }}" />
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700;900&display=swap" rel="stylesheet">
    <link
      href="https://fonts.googleapis.com/css2?family=Work+Sans&family=Reem+Kufi&family=Oswald&display=swap"
      rel="stylesheet"
    />
    <link
      href="https://fonts.googleapis.com/css2?family=DM+Serif+Display&family=Poppins&family=Roboto&display=swap"
      rel="stylesheet"
    />
    <style>
      .card1 a {
        text-decoration: none !important;
      }
    </style>
{% endblock %}

{% block content %}
<!-- <div style="background-color: blue; height:3rem"></div> -->
<div scroll = "no">
    <div class="prop">
      <img src="./../static/assets/prop.png"/>
    </div>
    <div class="heading">
      <div>
        <span class = "heading-text">Simps  ns Parody</span>
      </div>
      <div>
        <span class = "subtitle-text">CARTOON CHARACTER AUTOMATION</span>
      </div>
    </div>

```

```
<div class="container-fluid description" style="width: fit-content">
  <p
    class = "desc"
  >
  A deep learning project with the ability to recognise 20 characters from image,<br />
  and generate dialogues for episodes as long as needed from given context. <br />
  Efforts, Visualizations, Analysis, Working Demo and more!
</div>

<hr class="mt-5 mb-3" style="background-color: #ff9000; width: 45vw" />
<div class="row align-center cards">
  <div class="col-xl-4 col-lg-4 col-md-4 col-sm-12 my-3 p-0 mt-5" style="width: 38rem; ">
    <div class="column1" style="display: flex; justify-content: center">
      <div class="card1">
        <div class="imgBox1">
          <img src="./../static/assets/char-recog.png" alt="" />
        </div>
        <div class="cardBody1">
          <div class="top1">
            <div class="title">
              <span class="cardTitle">Character Recognition</span>
            </div>
            <p
              style="
                font-family: 'Roboto', sans-serif;
                font-weight: normal;
                color: #212121;
                font-size: 14px;
                margin-left: 1.3rem;
              "
            >
            Classfying 20 characters with
            DCNNs with the help of images.
            </p>
            <div class="footers">
              <a
                href="https://github.com/KushGabani/cartoon-character-automation/blob/master/training_phase_models/Simpsons_Character_Recognition_using_DCNNs.ipynb"
                target="_blank"
              >
                <img
                  src= "./../static/assets/GitHub-Mark-64px 3.png"
                  alt="GitHub-Mark-64px 3.png"
                  style="height: 2.5rem; width: 2.5rem; margin-left: 1rem"
                />
              </a>
              <a href="/character-recognition"
                ><button class="tryButton">TRY IT</button></a
```

```
              </div>
            </div>
          </div>
        </div>
      </div>

      <div
        class="col-xl-4 col-lg-4 col-md-4 col-sm-12 my-3 p-0 mt-5"
        style="width: 38rem;"
      >
        <div class="column1" style="display: flex; justify-content: center">
          <div class="card1">
            <div class="imgBox1">
              <img src="./../static/assets/script-gen.png" alt="" />
            </div>
            <div class="cardBody1">
              <div class="top1">
                <div class="title">
                  <span class="cardTitle">Script Generation</span>
                </div>
                <p
                  style="
                    font-family: 'Roboto', sans-serif;
                    font-weight: normal;
                    color: #212121;
                    font-size: 14px;
                    margin-left: 1.3rem;
                  "
                >
                Generating dialogues based on previous context using LTSMs.
                </p>
                <div class="footers">
                    <a href="https://github.com/KushGabani/cartoon-character-automation/blob/master/training_phase_models/Script_Generation_using_LSTMs.ipynb" target = "_blank
                        <img
                        src="./../static/assets/GitHub-Mark-64px 3.png"
                        alt="GitHub-Mark-64px 3.png"
                        style="height: 2.5rem; width: 2.5rem; margin-left: 1rem"
                        />
                    </a>
                    <a href="/script-generation">
                        <button class="tryButton">TRY IT</button>
                    </a>
                </div>
              </div>
            </div>
          </div>
        </div>
```

Character-recognition.html -

The web page for the first module of the project where DCNN is deployed for demo.

```
{% extends "base.html" %}

{% block title %}
  <title>Simpsons - Character-Recognition</title>
{% endblock %}

{% block css %}
    <link rel="stylesheet" href="{{ url_for('static', filename='css/Character-Recon.css') }}" />
    <link
      href="https://fonts.googleapis.com/css2?family=Work+Sans&family=Reem+Kufi&family=Oswald&family=Roboto&display=swap"
      rel="stylesheet"
    />
{% endblock %}

{% block content %}
<div class="main_container">
  <div>
    <img src="../static/assets/background.png" class="background" />
  </div>
  <div>
    <img src="../static/assets/donut.png" class="prop" />
  </div>
  <div class="container">
    <a href = "/" style = "text-decoration: none; color: black;">
    <div style = "width: fit-content; margin-top: 4rem;">
      <div style="display: flex; justify-content: start; align-items: center;">
        <img src="../static/assets/back.png" width="12" />
        <p style = "margin: 0 0.5rem; font-size: 20px;">Home</p>
      </div>
    </div>
    </a>
    <div class="content" id="shiftUp">
      <div class="reference">
        <a href="https://github.com/KushGabani/cartoon-character-automation/blob/master/training_phase_models/Simpsons_Character_Recognition_using_DCNNs.ipynb" target="_blank"
          <img
            src="../static/assets/GitHub-Mark-64px 3.png"
            alt="GitHub-Mark-64px 3.png"
            style="height: 2.4rem; width: 2.4rem"
          />  
        </a>
```

```
          <a
            href="https://www.kaggle.com/alexattia/the-simpsons-characters-dataset"
            style="color: #212121; text-decoration: none"
            target="_blank"
          >
            <span
              style="
                font-size: 2.5rem;
                font-family: 'Reem Kufi', sans;
                margin: 0 15px;
                padding-top: 5px;
              "
              >kaggle
            </span>
          </a>
        </div>

        <h1 class="title" style="color: #ffc804">Character Recognition</h1>

        <p class="description">
          Cartoon character recognition known as character recognition is used
          to identify cartoon characters in an image. From a total of around
          20 characters from the series, the correct one can be predicted
          using the image. Identify your character with just a click!
        </p>

        <div class = "buttonArena align-center" style = "position: relative;">
          <form id="uploadForm"
                enctype="multipart/form-data"
                method="post">
            <label for = "image-upload" class = "uploadBtn" >Try it</label>
            <input id="image-upload" type="file" name="file" accept="image/*"  style = "display: none;"/>
          </form>
          <span
            id="spinner"
            class="spinner-border text-dark"
            style="margin-left: 1rem; display: none"
          >
        </span>
        </div>
```

```
      <div class = "image-section" style="display: none">
        <div class = "img-preview">
          <div id="imagePreview">
          </div>
        </div>
        <div>
          <button type="button" class="predict" id="btn-predict"><span class = "btn-predict-text">Predict</span></button>
        </div>
      </div>

      <div style="margin: 1.5rem 0">
        <p id="result"></p>
      </div>
    </div>
  </div>
</div>


<div class="second-slide" style="margin-top: 10rem;">
  <div>
    <img src="../static/assets/background2.svg" class="background2" />
  </div>


  <div style = "display: flex; padding: 1.5rem; justify-content: center; align-items: center;">
    <b><h1 style = "color: black;">The Process</h1></b>
  </div>


  <div
    style="
      display: flex;
      justify-content: center;
      align-items: center;
      padding-top: 1.5rem;
    "
  >
    <div style="justify-content: center; padding: 5rem 0;">
      <img src="../static/assets/pages.png" class="prop2" />
    </div>
```

```
      </div>
      <div style="margin: 2rem;">
          <div style = "margin: 1rem 0;">
            <div style = "display: flex; justify-content: start; align-items: center;">
              <div><img src = "../static/assets/b1.png" width="75"/></div>
              <div style = "margin-top: 0.5rem; margin-left: 0.5rem;">
                <h3>How it's done?</h3>
              </div>
            </div>
            <hr/>
          </div>
          <div>
            <p class = "description">
              The image before passing on to the neural network, undergoes a preprocessing stage.
              The image is converted from BGR form to RGB and resized as a 100 x 100 pixel image.
              To classify the characters, a deep convolutional neural network consisting
              7 convolutional layers divided into 4 Pooling layers and then the extracted
              features from the convolutional layer are passed as an input to a dense 2 layer
              neural network.
            </p>
            <p class = "description">
              The dense neural network layers outputs probability for each
              character that could be in the image. charactes's class having the
              highest probability
              is taken as the predicted character.
            </p>
          </div>
        </div>
      </div>
    </div>
  </div>

  <div class="third-slide">
    <div>
      <img src = "../static/assets/background2.svg" class = "prop3a" />
    </div>

    <div class = "align-center third-heading">
      <h1 style = "font-family: Work Sans; font-weight: bold;">Model Statistics</h1>
    </div>

    <div class = "align-center">
      <div class = "inner-circle">
        <div
          class = "stat-circle align-center" style = "background-color: #4c4c4c;
                  filter: drop-shadow(7px 7px 50px rgba(0, 0, 0, 0.17));">
          <span style = "font-family: Oswald; color: #FFE266; font-size: 36px;">0.039</span>
        </div>
```

```
        <h4 style = "margin: 1rem 0;">Loss</h4>
      </div>

      <div class = "inner-circle">
        <div class="stat-circle align-center" style="background-color: #FFE266;
                        filter: drop-shadow(7px 7px 50px rgba(255, 226, 102, 0.43));">
          <span style="font-family: Oswald; color: #4c4c4c; font-size: 36px;">99.3%</span>
        </div>

        <h4 style = "margin: 1rem 0;">Accuracy</h4>
      </div>

      <div class = "inner-circle">
        <div class="stat-circle align-center" style="background-color: #E8E7E5;
                        filter: drop-shadow(7px 7px 50px rgba(9, 8, 8, 0.08));">
          <span style="font-family: Oswald; color: #7E7D7D; font-size: 36px;">0.99</span>
        </div>

        <h4 style = "margin: 1rem 0;">ROC AUC curve</h4>
      </div>
    </div>

    <div>
      <div class = "graphs align-center">
        <img src = "../static/assets/loss_graph.png" class = "graph"/>
        <img src = "../static/assets/accuracy_graph.png" class = "graph" />
      </div>
    </div>

    <div class = "align-center" style = "margin-bottom: 1.5rem">
      <a href = "https://tensorboard.dev/experiment/GyTX0fJIQzqpm4IVc2NQ0Q/" target = "_blank">
        <button type="button" class="predict" style="margin-top: 1.5rem;">
          <span class = "btn-predict-text">Visualize</span>
        </button>
      </a>
    </div>
  </div>

  {% endblock %}

  {% block scriptFooter %}
    <footer>
      <script type="text/javascript" src = "{{ url_for('static', filename='js/character-recon-script.js') }}"></script>
    </footer>
  {% endblock %}
```

Script-Generation.html -

The web page for the second module of the project where Sequence-to-Vector LSTM model is deployed for demo to generate new dialogues.

```
            <span
              style="
                font-size: 2.5rem;
                font-family: 'Reem Kufi', sans;
                margin-left: 1.5rem;
              "
              >kaggle
            </span>
          </a>
        </div>

        <h1 class="title" style="color: #ffc804">Script Generation</h1>

        <p class="description">
          Script Generation is a sequence to vector model that generates new dialogues
          with previous words as a context. Here this is implemented using LSTMs. The network
          has been trained with a total of around 80000 sequences with 50 words each.
          To use this functionality 50 words must be provided as a context for the model to complete
          and generate new dialogues. Create your dialogues now!
        </p>

        {% if(result == False) %}
        <div class="input-container">
            <form method="POST" id = "inputForm" action="/script-generation" style="display: flex;">
                <input type = "text" autocomplete="off" placeholder="100" class = "inputWord" id = "word" name = "word" />
                <input type = "text" autocomplete="off" placeholder="50 words as a context. . ." class = "inputText" id = "input" name = "input" />
                <button type="submit" name="submit" id = "submit" style = "display: none;"></button>
            </form>
        </div>

        <div class = "default-inputs">
            <div class="card1 col-4" id = "default1">
                <h3 class = "card-heading">Default Input I</h3>
                <p class="card-description">Marge: Well, leave it to good old Mary Bailey to finally step in and do something about that hideous genetic mutation. . .</p>
                <div class="go-corner" href="#">
                <div class="go-arrow">
                    →
                </div>
                </div>
            </div>

            <div class="card1 col-4" id = "default2" style = "margin-left: 4rem;">
                <h3 class = "card-heading">Default Input II</h3>
                <p class="card-description">Lenny_Leonard: It's too late to turn back, Moe. We've exchanged meaningful looks. No, you can still turn back!. . .</p>
                <div class="go-corner" href="#">
                <div class="go-arrow">
                    →
                </div>
```

```
                </div>
            </div>

            <div class="card1 col-4" id = "default3" style = "margin-left: 4rem;">
                <h3 class = "card-heading">Default Input III</h3>
                <p class="card-description">Carl_Carlson: Calm down, calm down! She doesn't know it's you. (SCREAMS) Hide! Hide! Quickly! Go! . .</p>
                <div class="go-corner" href="#">
                <div class="go-arrow">
                    →
                </div>
                </div>
            </div>
        </div>
        {% else %}
            <div class = "container script-output">
                    {% autoescape false %}
                        <p class = "css-typing"></p>
                        <p id = "hiddenOutput" style="display: none;">{{ result }}</p>
                    {% endautoescape%}
                <script>

                    let str = document.getElementById("hiddenOutput").innerHTML
                    var spans = '<span>' + str.split(' ').join('</span> <span>') + '</span>';
                    $(spans).hide().appendTo('.css-typing').each(function (i) {
                        $(this).delay(40 * i).css({
                            display: 'inline',
                            opacity: 0
                        }).animate({
                            opacity: 1
                        }, 100);
                    });

                </script>
            </div>

            <button onclick="handleDownloadClick(document.getElementById('hiddenOutput').innerHTML)" type="button" class="download" id="btn-download"><span class = "btn-downlo
        {% endif %}
      </div>
    </div>
  </div>

  <div class="third-slide">
    <div>
      <img src = "../static/assets/background2.svg" class = "prop3a" />
    </div>

    <div class = "align-center third-heading">
      <h1 style = "font-family: Work Sans; font-weight: bold;">Model Statistics</h1>
```

```
        </div>

        <div class = "align-center">
          <div class = "inner-circle">
            <div
              class = "stat-circle align-center" style = "background-color: #4c4c4c;
                      filter: drop-shadow(7px 7px 50px rgba(0, 0, 0, 0.17));">
              <span style = "font-family: Oswald; color: #FFE266; font-size: 36px;">0.098</span>
            </div>

            <h4 style = "margin: 1rem 0;">Loss</h4>
          </div>

          <div class = "inner-circle">
            <div class="stat-circle align-center" style="background-color: #FFE266;
                              filter: drop-shadow(7px 7px 50px rgba(255, 226, 102, 0.43));">
              <span style="font-family: Oswald; color: #4c4c4c; font-size: 36px;">97.1%</span>
            </div>

            <h4 style = "margin: 1rem 0;">Accuracy</h4>
          </div>

          <div class = "inner-circle">
            <div class="stat-circle align-center" style="background-color: #FFE266;
                              filter: drop-shadow(7px 7px 50px rgba(255, 226, 102, 0.43));">
              <span style="font-family: Oswald; color: #4c4c4c; font-size: 36px;">0.04</span>
            </div>

            <h4 style = "margin: 1rem 0;">MAE</h4>
          </div>
        </div>
        <div>
          <div class = "graphs align-center">
            <img src = "../static/assets/scipt-accuracy-graph.png" class = "graph"/>
            <img src = "../static/assets/script-loss-graph.png" class = "graph" />
          </div>
        </div>

        <div class="align-center" style="margin: 1.5rem 0;">
          <a href = "https://tensorboard.dev/experiment/JOWxat2WSXWDCnDYEFvM7Q/" target = "_blank">
            <button type="button" class="predict">
              <span class = "btn-predict-text">Visualize</span>
            </button>
          </a>
        </div>
      </div>
    {% endblock %}
```

Home.css – The CSS styling for Home.html

```css
html, body {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    overflow-x: hidden;
}

.align-center {
    display: flex;
    justify-content: center;
    align-items: center;
}

.heading {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    margin: 12rem 0 2.5rem 0;
  }

.heading-text {
    font-family: Poppins;
    font-style: normal;
    font-weight: 500;
    font-size: 45px;
    line-height: 67px;
    display: flex;
    align-items: center;
    letter-spacing: 0.065em;
    color: #000000;
}

.subtitle-text {
    font-family: Poppins;
    font-style: normal;
    font-weight: 200;
    font-size: 16px;
    line-height: 24px;
    display: flex;
    align-items: center;
    letter-spacing: 0.3em;
    color: #000000;
}
```

```css
.cards {
  margin: 3rem;
  background-image: url('./../assets/home_background.png');
  background-repeat: repeat;
  background-size: auto;
}

  .card1 {
    width: 65%;
    display: block;
    border-radius: 15px;
    box-shadow: -25px -25px 50px 2px rgba(0, 0, 0, 0.10),
      25px 25px 50px 2px rgba(0, 0, 0, 0.10);
  }

  .cardTitle {
    font-family: "DM Serif Display", serif;
    margin: 1.4rem 0 1rem 0.5rem;
    font-size: large;
    font-weight: normal;
  }

  .cardDescription {
    font-family: "Roboto", sans-serif;
    font-weight: normal;
    color: black;
    font-size: 16;
    margin-left: 1.3rem;
  }

  .card1 a {
    color: #575555;
  }

  .imgBox1 {
    margin: 0;
    padding: 0;
    border-radius: 15px 15px 0 0;
    height: 135px;
    width: 100%;
  }
  .imgBox1 img {
    margin: 0;
    padding: 0;
    width: 100%;
    height: 100%;
    border-radius: 15px 15px 0px 0px;
  }
```

```css
.tryButton {
  outline: none;
  width: 95px;
  height: 32px;
  padding: 0px;
  border-radius: 5px;
  border: none;
  background: #ff9000;
  margin-right: 2rem;
  color: white;
  font-size: small;
}

.title {
  color: black;
  text-align: center;
  display: flex;
  margin-top : 2rem!important;
  margin-bottom : 1rem!important;
}

.cardBody1 {
  display: block;
  background: #fff;
  border-radius: 0 0 15px 15px;
  height: fit-content;
  margin-top: -10px;
  padding: 0;
}

.cardBody1 .title {
  margin: 0.8rem 0 0 0.8rem;
  font-weight: 500;
  line-height: 0.8rem;
}

.cardBody1 p {
  margin: 0 0.8rem;
  font-weight: lighter;
  color: #686868;
  font-size: 0.9rem;
  font-family: Cambria, Cochin, Georgia, Times, "Times New Roman", serif;
}

.card1 a {
  text-decoration: none;
  color: black;
}
```

```css

.card1:hover {
  transition-property: box-height, margin-top, box-shadow;
  transition-duration: 0.4s;
  transition-timing-function: ease-out;
  margin-top: -20px;
  height: 20%;
  visibility: visible;
  box-shadow: -10px -10px 30px 2px rgba(0, 0, 0, 0.15),
    10px 10px 30px 2px rgba(0, 0, 0, 0.15);
}

.description {
  display: flex;
  margin: 3rem 0 ;
  justify-content:  center;
  align-items: center;
}

.desc {
  font-family: Roboto;
  font-style: normal;
  font-weight: normal;
  font-size: 18px;
  display: flex;
  align-items: center;
  text-align: center;
  letter-spacing: 0.02em;
  color: #000000;
}

.footers {
  margin-bottom: 2rem;
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-top: 6rem;
  margin-left: 1rem;
  margin-right: 0.5rem;
}

.prop {
  width: 20rem;
  position: absolute;
  top: -1.3%;
  left: 39%;
}

```

Character-Recon.css – The CSS styling attributes for character-recognition.html

```css
body {
  margin: 0;
  padding: 0;
  background-attachment: fixed;
  background-repeat: no-repeat;
}

.align-center {
  display: flex;
  justify-content: center;
  align-items: center;
}

.container {
  height: 100vh;
}

.second-slide {
  height: 100vh;
  position: relative;
}

.third-slide {
  height: 100vh;
  position: relative;
}

.background {
  position: absolute;
  right: 15rem;
  bottom: 0;
  width: 35rem;
}

.background2 {
  position: absolute;
  left: 0;
  bottom: 0;
  width: 35rem;
  z-index: -1;
}

.prop {
  position: absolute;
  width: 16rem;
  left: 55rem;
  top: 0;
}

.prop2 {
  width: 50rem;
  margin-right: 10rem;
}

.prop3a {
  position: absolute;
  left: 0;
  top: 0;
  transform: rotateX(180deg);
  width: 35rem;
  z-index: -1;
}

.content {
  padding-top: 9rem;
  width: fit-content;
}

.third-heading {
  padding: 2.5rem;
}

.description {
  font-size: large;
  margin: 0.5rem 0;
  font-weight: normal;
  width: 40rem;
  height: auto;
  font-family: "Roboto";
}

.reference {
  display: flex;
  justify-content: left;
  align-items: center;
  margin-top: 1.5rem;
  margin-bottom: 0.5rem;
  width: fit-content;
}

.subtitle {
  font-family: Poppins;
  margin-bottom: 0;
  font-weight: bold;
}
```

```css
.title {
  font-family: "Work Sans";
  font-weight: bolder;
  width: fit-content;
}

.buttonArena {
  width: fit-content;
  display: flex;
}

.uploadBtn {
  margin: 2.5rem 0;
  width: 180px;
  height: 44px;
  display: block;
  text-align: center;
  padding: 0.8rem;
  border-radius: 5px;
  letter-spacing: 0.2em;
  font-family: "Work Sans", sans;
  box-shadow: 10px 10px 20px rgba(120, 120, 0, 0.18);
  border: none;
  background: #212121;
  margin-right: 5px;
  color: white;
  font-size: large;
  font-weight: bold;
  cursor: pointer;
}

#result {
  font-family: "Work Sans";
  font-weight: bold;
  font-size: larger;
}

.img-preview {
  width: 130px;
  height: 250px;
  position: relative;
  margin: 1rem 0;
}

.img-preview>div {
  width: 100%;
  height: 100%;
  background-size: 145px 175px;
  background-position: center;
}

input[type="file"] {
  display: none;
}

.predict {
  width: 180px;
  height: 44px;
  padding: 0.5rem;
  border-radius: 5px;
  box-shadow: 10px 10px 20px rgba(236, 236, 3, 0.18);
  border: none;
  background: #ffc804;
  margin-right: 5px;
  cursor: pointer;
}

.btn-predict-text {
  color: black;
  font-size: larger;
  font-weight: bold;
  font-family: "Work Sans", sans;
}

.stat-circle {
  width: 12rem;
  height: 12rem;
  border-radius: 50%;
  text-align: center;
}

.inner-circle {
  margin: 0 5rem;
  display: flex;
  flex-direction: column;
  align-items: center;
}

.graphs {
  margin: 5rem 0;
}

.graph {
  margin: 2rem 3rem;
}
```

Script-Gen.css – The CSS style attributes for script-generation.html

```css
.html,
.body {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

.align-center {
    display: flex;
    justify-content: center;
    align-items: center;
}

.inputText {
    width: 70vw;
    height: 6rem;
    border: none;
    overflow: hidden;
    font-family: Roboto;
    font-size: 40px;
    font-weight: bold;
    letter-spacing: 0.2rem;
    border-bottom: 2px solid #ffc804;
}

.inputText::placeholder {
    letter-spacing: 0;
    font-style: italic;
    opacity: 0.3;
    color: lightgray;
}

.input-container:hover .inputText::placeholder{
    opacity: 0.9;
    transition: opacity 0.3s ease-out;
}

.inputText:focus {
    outline: none;
}

.inputWord {
    margin-right: 5rem;
    width: 7rem;
    border: 0;
    font-family: Roboto;
    font-size: 34px;
    font-weight: bold;
    letter-spacing: 0.2rem;
    border-bottom: 2px solid #292929;
}

.inputWord::placeholder {
    opacity: 0.5;
    color: lightgray;
}

.inputWord:focus {
    outline: none;
}

.input-container {
    display: flex;
    margin: 5rem 0;
    justify-content: center;
    align-items: center;
}

.default-inputs {
    display: flex;
    justify-content: center;
    height: fit-content;
    margin-bottom: 5rem;
}

.card1 {
    display: block;
    position: relative;
    max-width: 350px;
    height: fit-content;
    background-color: #fcfcfc;
    box-shadow: 10px 10px 15px rgba(0, 0, 0, 0.08);
    border-radius: 4px;
    padding: 1rem 3rem;
    text-decoration: none;
    z-index: 0;
    overflow: hidden;
}

.card1::before {
    content: "";
    position: absolute;
    z-index: -1;
    top: -16px;
    right: -16px;
    background: #262626;
```

```
        border-radius: 50px;
        transform: scale(1);
        transform-origin: 50% 50%;
        transition: transform 0.25s ease-out;
    }

    .card1:hover::before {
        transform: scale(21);
        cursor: pointer;
    }

    .card1:hover .card-description {
        transition: all 0.3s ease-out;
        color: rgba(255, 255, 255, 0.8);
    }

    .card1:hover .card-heading {
        transition: all 0.3s ease-out;
        color: #ffc804;
    }

    .card-heading {
        text-decoration: none;
        color: #262626;
        font-size: 24px;
        line-height: 24px;
        font-weight: 700;
        font-family: Roboto;
        margin-bottom: 0.7rem;
    }

    .card-description {
        text-decoration: none;
        font-size: 14px;
        font-weight: 500;
        font-family: Roboto;
        line-height: 20px;
        color: #3a3a3a;
    }

    .go-corner {
        display: flex;
        align-items: center;
        justify-content: center;
        position: absolute;
        width: 32px;
        height: 32px;
        overflow: hidden;
        top: 0;
        right: 0;
        background-color: #1e1f1f;
        border-radius: 0 4px 0 32px;
    }

    .go-arrow {
        margin-top: -0.6rem;
        margin-right: -0.5rem;
        color: white;
        font-family: courier, sans;
    }

    .script-output {
        height: fit-content;
        margin: 5rem 0;
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
        padding-top: 2rem;
        padding-bottom: 1rem;
        border: 3px solid #ffc804;
    }

    .css-typing {
        font-family: Roboto;
        font-weight: 500;
        color: #292929;
        font-size: 24px;
        letter-spacing: 0.08rem;
    }

    .download {
        width: 180px;
        height: 44px;
        padding: 0.5rem;
        border-radius: 5px;
        box-shadow: 10px 10px 20px rgba(236, 236, 3, 0.18);
        border: none;
        background: #ffc804;
        margin-right: 5px;
        margin-bottom: 3rem;
        cursor: pointer;
    }
    .btn-download-text {
        color: black;
        font-size: larger;
```

```
    font-weight: bold;
    font-family: Roboto;
    letter-spacing: 0.1rem;
    font-family: "Work Sans", sans;
  }

.main_container {
    margin-left: -10rem;
}

.prop {
    position: absolute;
    width: 35rem;
    right: 18rem;
    top: 13rem;
}

.content {
    padding-top: 3rem;
    width: fit-content;
}

.third-slide {
    /* height: 100%; */
    position: relative;
}

.third-heading {
    padding: 2.5rem;
}

.prop3a {
    position: absolute;
    left: 0;
    top: 0;
    transform: rotateX(180deg);
    width: 35rem;
    z-index: -1;
  }

.description {
    font-size: large;
    margin: 2rem 0;
    font-weight: normal;
    width: 55rem;
    height: auto;
    font-family: "Roboto";
}
```

```
.reference {
    display: flex;
    justify-content: left;
    align-items: center;
    margin-top: 1.5rem;
    margin-bottom: 0.5rem;
    width: fit-content;
}

.subtitle {
    font-family: Poppins;
    margin-bottom: 0;
    font-weight: bold;
}

.title {
    font-family: "Work Sans";
    font-weight: bolder;
    width: fit-content;
}

.stat-circle {
    width: 12rem;
    height: 12rem;
    border-radius: 50%;
    text-align: center;
}

.inner-circle {
    margin: 0 5rem;
    display: flex;
    flex-direction: column;
    align-items: center;
}

.graphs {
    margin-top: 5rem;
}

.graph {
    margin: 2rem 8rem;
}


.predict {
    width: 180px;
    height: 44px;
    padding: 0.5rem;
```

```
    box-shadow: 10px 10px 20px rgba(236, 236, 3, 0.18);
    border: none;
    background: #ffc804;
    margin-right: 5px;
    cursor: pointer;
}

.btn-predict-text {
    color: black;
    font-size: larger;
    font-weight: bold;
    font-family: "Work Sans", sans;
}


@media screen and (max-width: 1024px) {
    .prop {
        display: none;
    }
}
```

character-recon-script.js – Logic for the module character-recognition.html

```
$(document).ready(function() {
    console.log("ready!")
     $('.image-section').hide();
     $('#spinner').hide();
     $('#result').hide();

     function readURL(input) {
         if(input.files && input.files[0]) {
             var reader = new FileReader();
             reader.onload = function(e) {
                 $('#imagePreview').css('background-image', 'url(' + e.target.result + ')');
                 $('#imagePreview').hide();
                 $('#imagePreview').fadeIn(650);
             }
             reader.readAsDataURL(input.files[0]);
         }
     }

     $('#image-upload').change(function() {
        document.getElementById("shiftUp").style.transition = "all 0.3s"
        document.getElementById("shiftUp").style.paddingTop = 0;
        $('.image-section').show();
        $('#btn-predict').show();
        $('#result').text('');
        $('#result').hide();
        readURL(this);
     });

     $('#btn-predict').click(function() {
         var form_data = new FormData($('#uploadForm')[0]);
         $(this).hide();
         $('#spinner').show();
        console.log("Reached ajax post.")
         $.ajax({
            type: 'POST',
            url: window.location.origin + '/predict',
            data: form_data,
            contentType: false,
            cache: false,
            processData: false,
            async: true,
            success: function(data) {
                $('#spinner').hide();
                $('#result').fadeIn(600);
                $('#result').text("Looks like it's " + data + " here.");
                console.log('Success!');
            }
         });
```

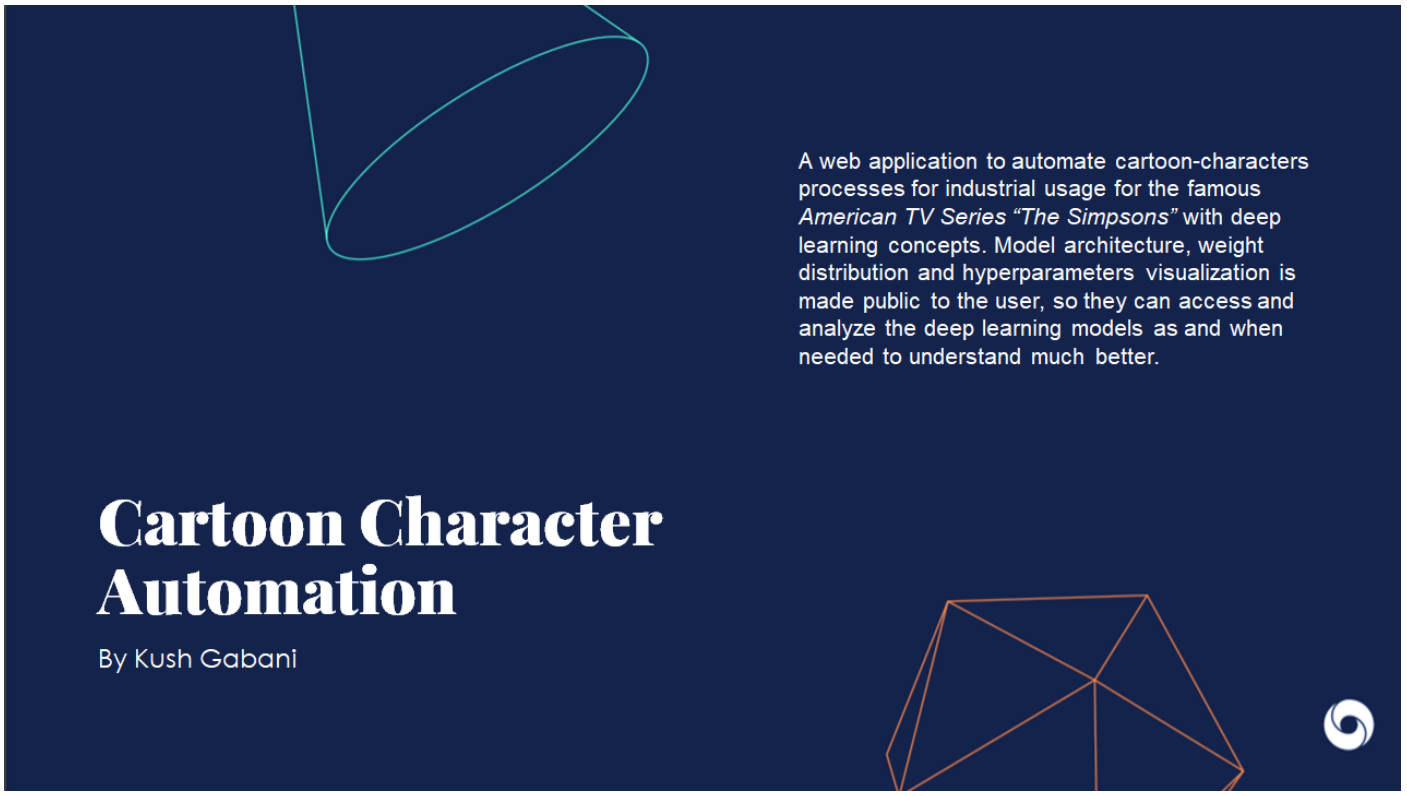
script-gen.js – Logic for script-generation.html

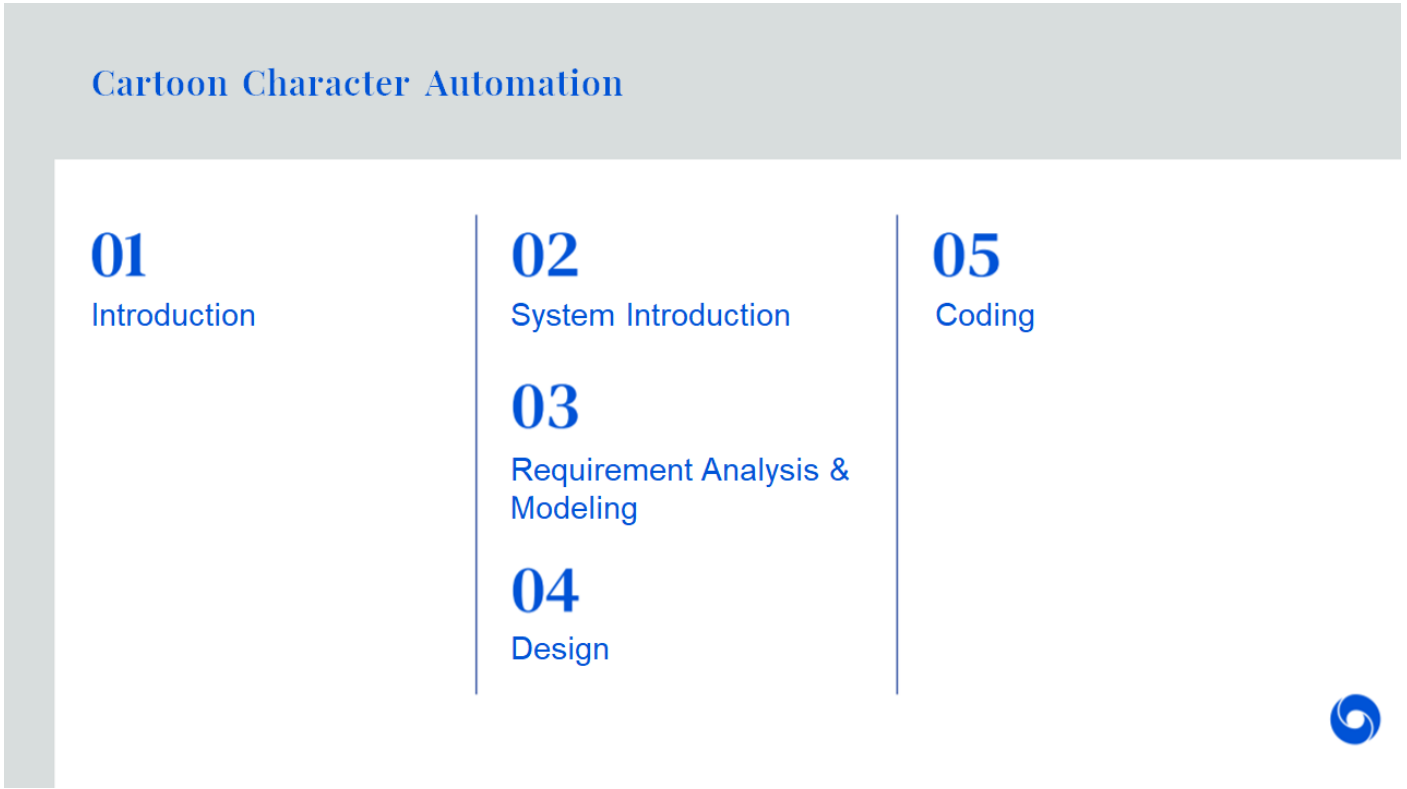
```
const handleDownloadClick = (text) => {
    text = text.replaceAll("<br>", "")
    var element = document.createElement('a');
    element.setAttribute('href', 'data:text/plain;charset=utf-8,' + encodeURIComponent(text));
    element.setAttribute('download', "./generated_script.txt");
    element.style.display = 'none';

    document.body.appendChild(element);
    element.click();

    document.body.removeChild(element);
}

try {
    document.getElementById("input").addEventListener("keyup", function(event) {
        if (event.keyCode === 13) {
            let input = document.getElementById("input").value;
            submitInput(input);
        }
    });

    document.getElementById("default1").addEventListener("click", function(event) {
        console.log("Default 1!");
        let input = `Marge: Well, leave it to good old Mary Bailey to finally step in and do something about that hideous genetic mutation. <br/>
                     Homer: Mary Bailey well if I was governor i'd sure find better things to do with my time. <br/>
                     Marge: Like what? homer like getting,`
        submitInput(input);
    })

    document.getElementById("default2").addEventListener("click", function(event) {
        console.log("Default 2!");
        let input = `Lenny_Leonard: It's too late to turn back, Moe. We've exchanged meaningful looks. <br/>
                    Moe_Szyslak: No, you can still turn back! The point of no return is the whispered huddle. <br/>
                    Moe_Szyslak: Oh God, oh God! <br/>
                    Homer_Simpson: Ahhh, that is so much better than hospital beer. <br/>
                    Lenny_Leonard: Homer, where you been the last`
        submitInput(input);
    })

    document.getElementById("default3").addEventListener("click", function(event) {
        console.log("Default 3!");
        let input = `Carl_Carlson: Calm down, calm down! She doesn't know it's you. <br/>
                    Carl_Carlson: (SCREAMS) Hide! Hide! <br/>
                    Moe_Szyslak: Uh, hello? Oh, sure, Liser, your Dad's right here.<br/>
                    Lisa_Simpson: (ON PHONE) Dad? Did you just call? <br/>
                    Homer_Simpson: Uh, yeah. Hey, listen, your mom thinks that maybe you and I should have dinner together, sometime`
        submitInput(input);
```

```
    })

    let submitInput = (input) => {
        console.log(input);
        document.getElementById("input").value = input;
        document.getElementById("submit").click();
    }

}
catch {

}
```

Cartoon Character Automation Presentation

(for educational purposes and understanding the project)





1 Introduction

About the Project !

- Project Title –
  Cartoon-character automation using Deep learning
- Project Description –
  A deep learning utility to recognize characters and generate dialogues for episodes.
- Internal Guide –
  Yesha Ma'am
- Technology Used –
  - Front-end tools –
    HTML, CSS, Javascript, Jquery, Bootstrap
  - Back-end tools –
    Flask
    Tensorflow | Keras
  - Visualization tool –
    Tensorboard

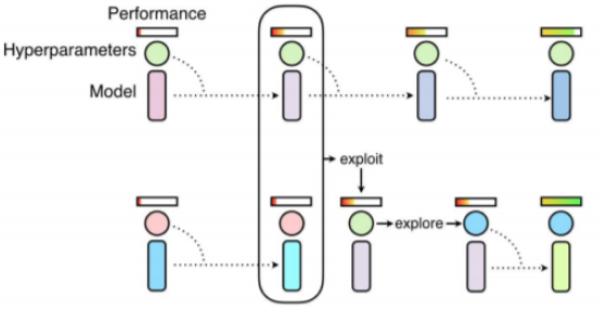
2
System Introduction

System Definition

- A web application to automate cartoon-characters processes for industrial usage for the famous American TV Series “The Simpsons” with deep learning concepts.
- The application consists of two modules - Character Recognition and Script Generation. The Character Recognition module recognises and classifies images from the wide variety of characters in the series, 20 characters to be specific; with the help of Deep Convolutional Neural Networks (DCNN) with the relatively high accuracy of 99.7%.
- The Script Generation module generates n number of new dialogues for the episode given a few words as a context and beginning; with sequence-to-vector models consisting of Long Short Term Memory (LSTM) Cells, summing all up, the training phase resulted in a low Mean Absolute Error (MAE) of 0.03
- Model architecture, weight distribution and hyperparameters visualization is made public to the user, so they can access and analyze the deep learning models as and when needed to understand much better.

System Objective

- The project aims at automating various processes in industrial areas like analysis and even on a contextual level.
- The Character Recognition module should be able to accurately classify among the 20 characters using images as an input.
- The Script Generation module is capable of generating as many new dialogues as needed given a context of around 10-20 words
- Model architectures and visualization of both the trained deep learning can be used for analysis of the training phase of the models. The recorded log file data visualisation can be also used for hyper-parameter tuning to further improve the model's accuracy and decrease the loss or educational purposes like how the model trains? Or analysing the model's architecture.

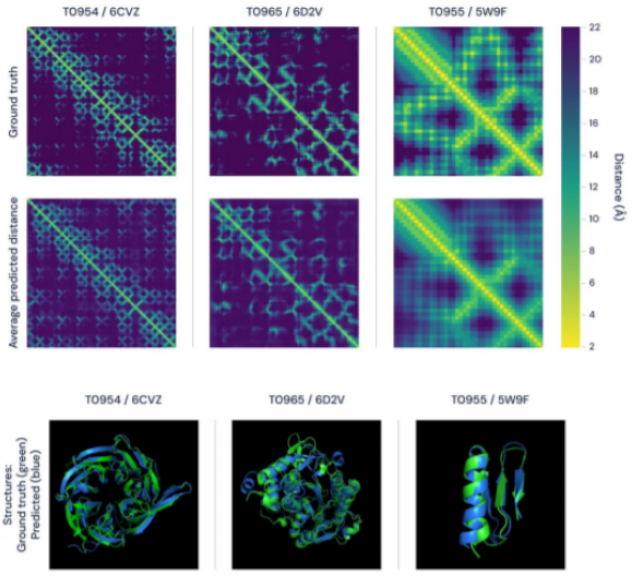


System Scope

Character Recognition

- The Character Recognition module can be modified into a meta-application or an API for another application like calculating "screen time (appearance)" of characters in each episode.
- The DCNN of the Character Recognition can also be used as a base model for Faster RCNN (Regional Convolutional Neural Network) in Object Detection application. Image segmentation can also be implemented using this DCNN as a base model. Furthermore, The same DCNN architecture can be modified slightly to be used as a high accuracy discriminator in GANs (General Adversarial Networks) to generate new Characters.

continued...



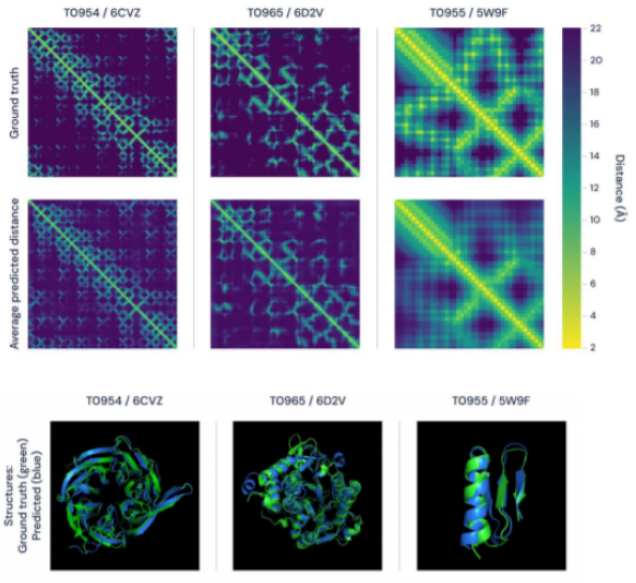
System Scope

Script Generation

→ The Script Generation module can further be fine-tuned to generate an entire new series with new characters replacing the old ones for example, The Simpsons but in Hindi using Google Translate API. The generated script can be further be used as an input sequence for Sentiment-Analysis application that can then classify the emotion of the generated script

Visualisation

→ Model Architecture Analysis and understanding the deep neural networks at a relatively low-level. This can be used as a stage to then analyze a model and write a research, thesis or dissertation on these models.



Hardware & Software Requirements

HARDWARE REQUIREMENTS

TRAINING PHASE (Google Colab)

- Processor - Xeon Processor @ 2.3Ghz
- RAM - 12 GB DDR5
- SSD - 64 GB
- GPU - Tesla K80 with 2496 CUDA cores

DEPLOYMENT PHASE

- Processor - Intel i5 @ 1.4Ghz
- RAM - 4 GB DDR3
- SSD - 64 GB

SOFTWARE REQUIREMENTS

Operating System - Windows 7 / OSX Catalina or above

Training Environment - Google Colab with GPU

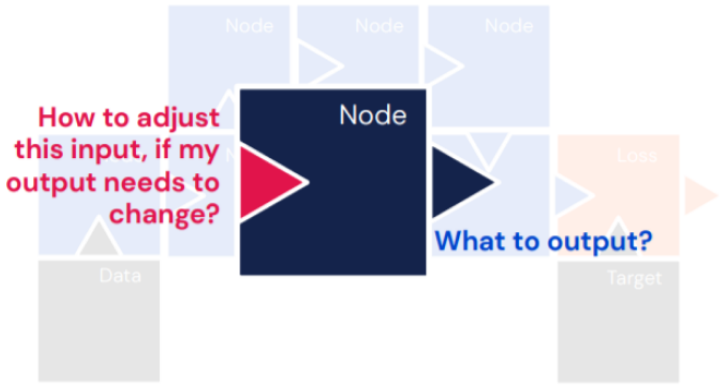
Front end tools - HTML, CSS, Javascript

Back end tools - Flask, Tensorflow (v2.x>=), Keras

3

Requirement Analysis & Modeling

Expected working of the system

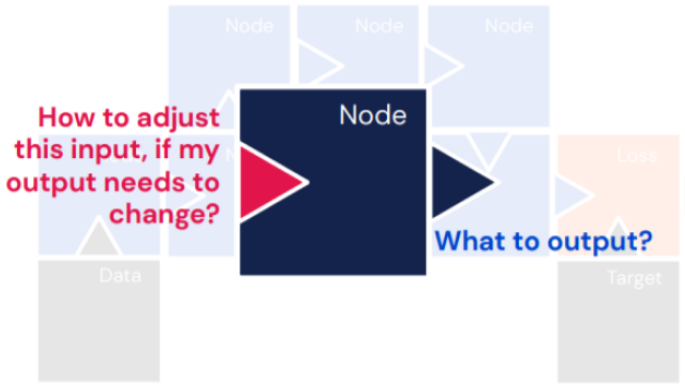


HOME PAGE

- This is the default page that first loads when the flask server is in deployment. When run locally, the web application can be accessed with Localhost PORT number 5000
- The home page will define the route to the two main modules of the project; namely, Character Recognition and Script Generation.
- When clicked on either of them, the web application is redirected to that specific module where the deployed model functions.
- Code for both the modules can be accessed by clicking on the github icons on either of the modules.

CONTINUED...

Expected working of the system

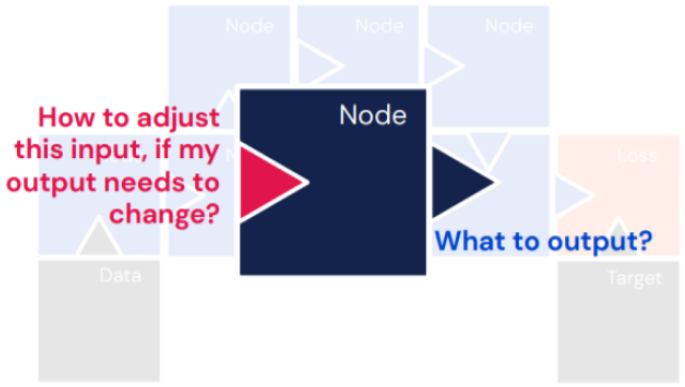


CHARACTER RECOGNITION PAGE

- The webpage where the DCNN model for Character Recognition is deployed. The user should be able to upload any cartoon image that needs to be classified among the 20 characters.
- The image is then uploaded to the server using JQuery and AJAX. To be able to pass the image to the DCNN, it first needs to be preprocessed. The image is read into an nd-array then resized to 100 x 100 resolution.
- The DCNN then should be able to classify and display the output on the web page accurately.
- The user can also visualize the training phase of the DCNN Model and its architecture, analyze the weight distribution of internal parameters and tune hyper-paramers for educational purposes. Model Statistics of the DCNN should also be viewed by the user for reference.

CONTINUED...

Expected working of the system

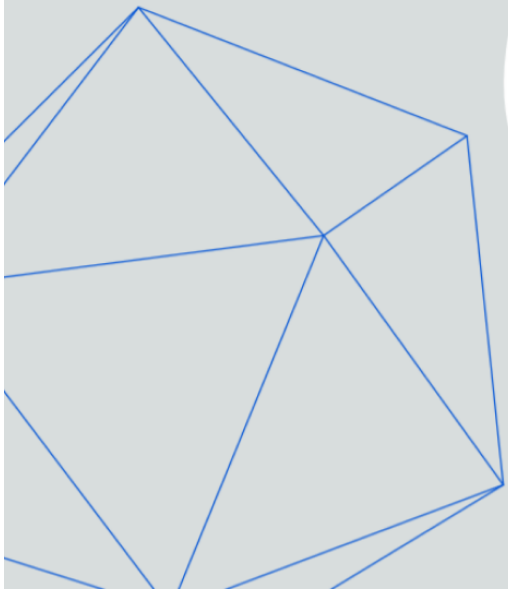


SCRIPT GENERATION PAGE

- The webpage where the Sequence-to-Vector LSTM Model for Script Generation is deployed. The user should be able to input the number of words (default = 100) to be generated by the Neural Network and a few words as a context for the dialogues to be generated, generally 30-40 words.
- If the user does not want to enter any context then three default contexts are provided for the input.
- The user should also be able to visualize the training phase of the Sequence-to-Vector LSTM model in graphs, flowcharts and tune hyper-parameters. This should also be enough for a user to base their dissertation on.
- Model Statistics of the deployed model should also be accessible to the user.
- The user should also be able to download the generated text into a text file for their personal future use.

Convolutional Neural Networks for Image Recognition

In the past decade , convolutional neural networks have revlutionalised computer vision.
Futhur coming up is the closer look at CNN architectures built in the project.



How can we feed images to a neural network?

Images cannot be fed as input to vanilla neural networks becuase of the way it flattens. Due to pixels flattening, even if the image is changed slightly the network might output something completely different. Thus, because of locality and translation invariance simple neural network cannot be used for images. To work with images, important feature that define the image must be extracted.

Convolutional Neural Network

1. FILTERS

In convolutional layers, we instead have a collection of smaller feature detectors called convolutional filters which we individually slide along the entire image and perform the same weighted sum operation as before, on each subregion of the image. Essentially, for each of these small filters, we generate a map of responses–called an activation map–which indicate the presence of that feature across the image.

2. NON LINEARITY

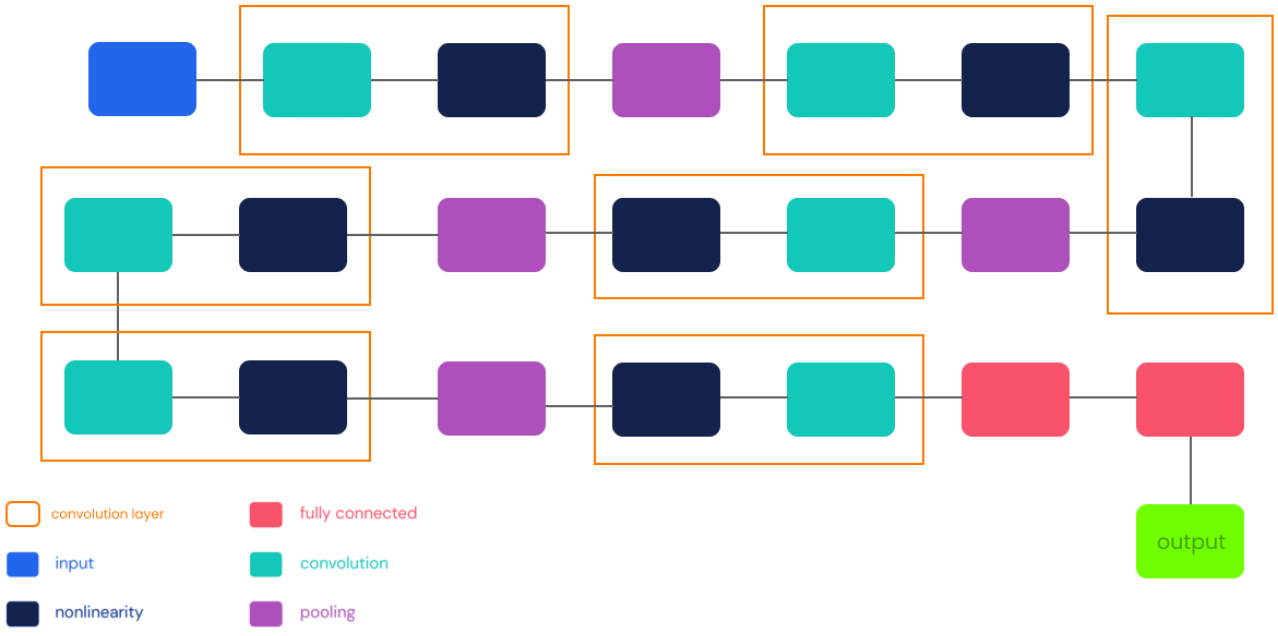
non-linearity layer in a convolutional neural network consists of an activation function that takes the feature map generated by the convolutional layer and creates the activation map as its output. The activation function is an element-wise operation over the input volume and therefore the dimensions of the input and the output are identical. Generally, ReLu is used as a non linearity in CNNs

3. POOLING

The pooling operation is used to downsample the activation maps, usually by a factor of 2 in both dimensions. The advantage of pooling is that it gives us a way to compactify the amount of data without losing too much information, and create some invariance to translational shift in the original image. The operation is also very cheap since there are no weights or parameters to learn.

Filter and Non Linearity are stacked together to make 1 convolution layer

Character Recognition module's DCNN Architecture



Implementational Training Details

Loss function

Categorical Cross-entropy

Hyper-parameters

Optimizer

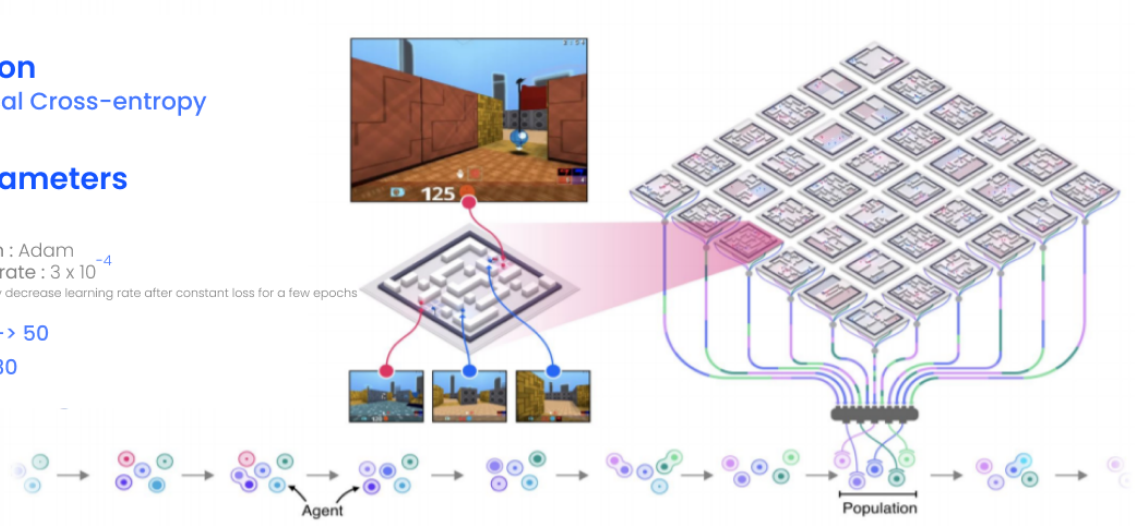
Algorithm : Adam

Learning rate : 3×10^{-4}

*automatically decrease learning rate after constant loss for a few epochs

Batch Size -> 50

Epochs -> 30



Visualization

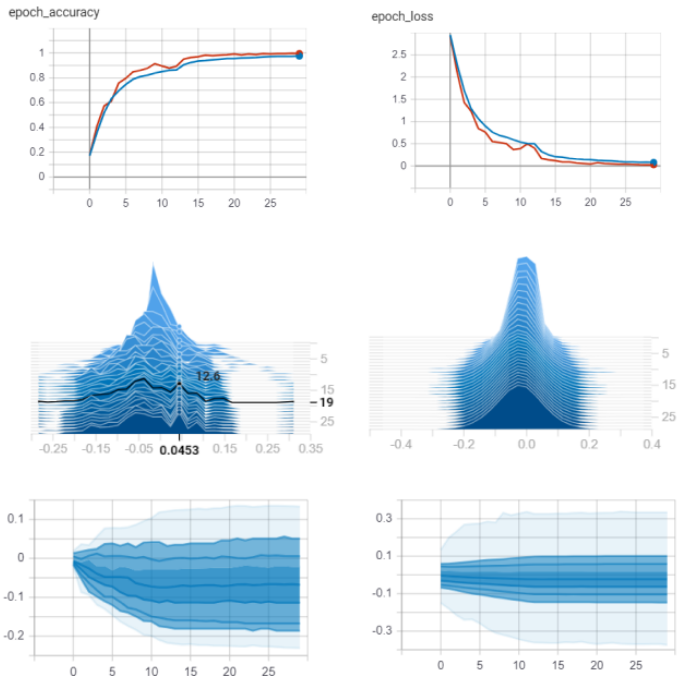
DCNN Model Architecture, weight distribution and training history

Visualization

TRAINING HISTORY

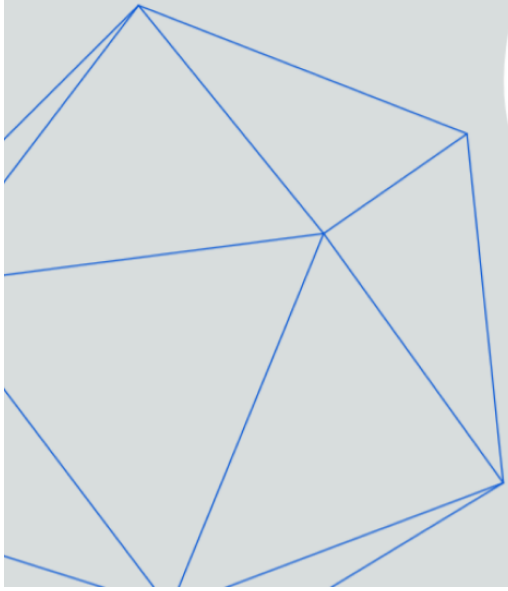
CONV. HISTOGRAMS

WEIGHT DISTRIBUTIONS



Here we focus on sequential data and how deep learning methods have been adapted to process this particular type of structure. We will start by introducing some fundamentals of sequence modeling including common architectures designed for this task such as LSTMs. We will then move on to sequence-to-vector decoding.

Long Short Term Memory Cells & Sequencial data



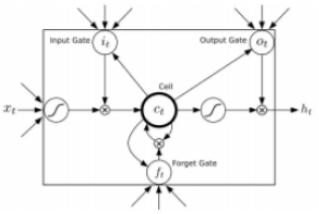
Why other architectures doesn't work well with sequential data?

The models that have been discussed so far are not built for sequential data because those models does not take order into account. Each sequence can be of variable length, For example, we cannot have a feedforward neural network with input layer being dynamic. On the other hand, the order might also contain extra information other than the extracted features from a single data.

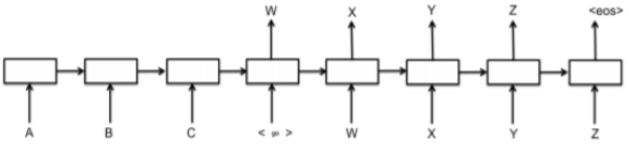
Sequences and Recurrent Networks

(Marta Garnelo)

- Almost all data is sequential : text, DNA, video, audio
- How can we process such data using machine learning
- Fundamentals of sequence modeling including Recurrent Neural Networks and Long-Short Term Memory (LSTMs)
- Mapping sequences to sequences as in machine translation



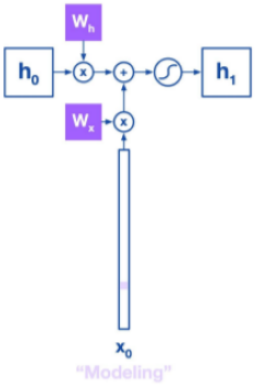
LSTM cell (Long-short term memory, Hochreiter & Schmidhuber, Neural Computation 1997)



Sequence to sequence learning with neural networks (Sutskever et. al, NIPS 2014)

Recurrent Neural Networks (RNNs)

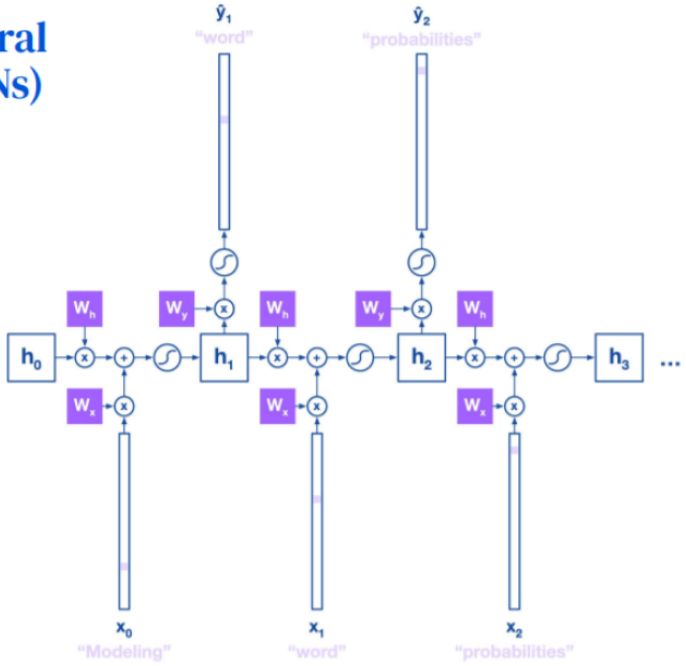
Persistent state variable **h** stores information from the context observed so far.



$$\mathbf{h}_t = \tanh(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t)$$

Elman (1991)

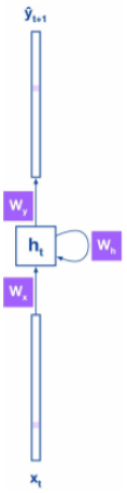
Recurrent Neural Networks (RNNs)



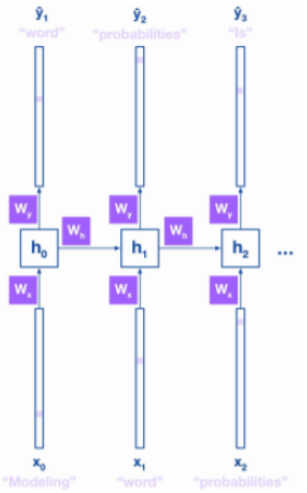
Elman (1991)

Recurrent Neural Networks (RNNs)

Weights are shared over time steps



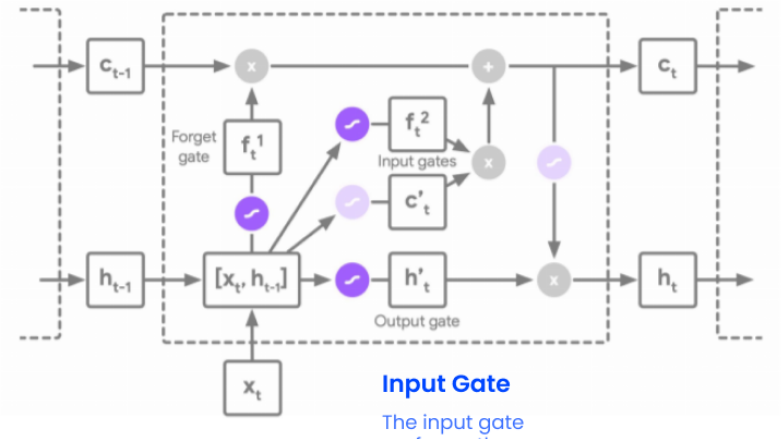
RNN



RNN rolled out over time

Long Short-Term Memory (LSTM) networks

LSTM state update



Forget Gate

The forget gate decides which information needs attention and which can be ignored.

Input Gate

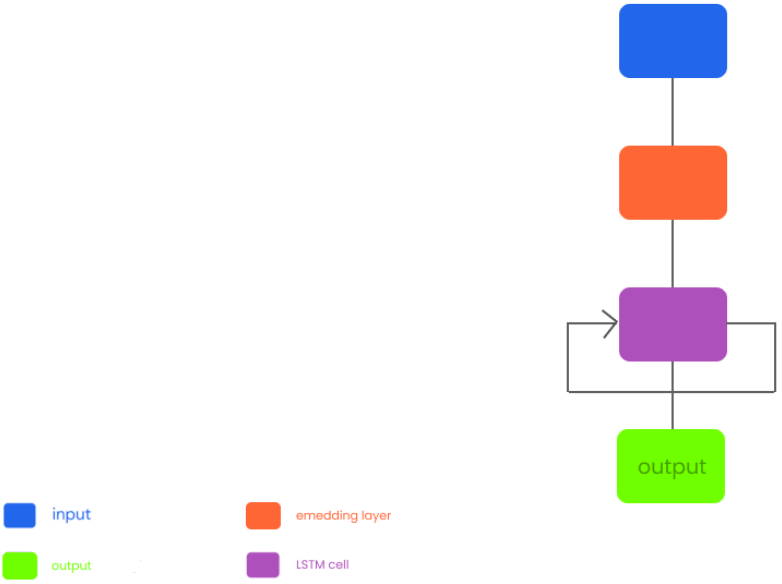
The input gate performs the operations to update the cell status.

Output Gate

The output gate determines the value of the next hidden state. This state contains information on previous inputs.

Hochreiter (1997), Gers et al. (1999)

Script Generation module's S2V LSTM Architecture



Implementational Training Details

Loss function

Categorical Cross-entropy

Hyper-parameters

Optimizer

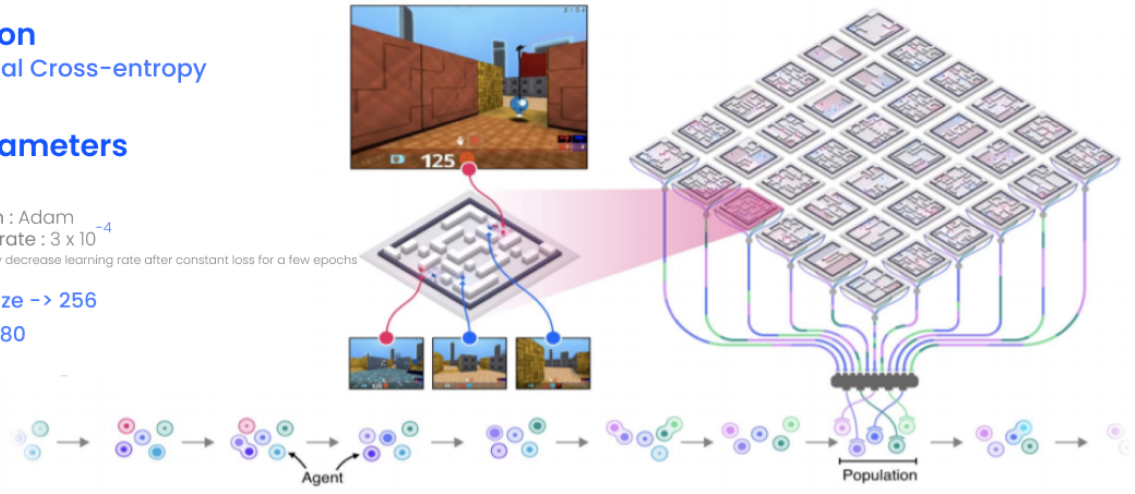
Algorithm : Adam

Learning rate : 3×10^{-4}

*automatically decrease learning rate after constant loss for a few epochs

LSTM Cell Size -> 256

Epochs -> 180



Visualization

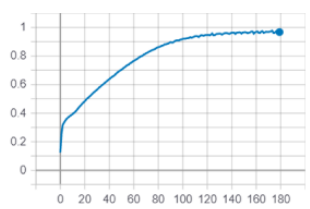
LSTM Model Architecture, weight distribution and training history

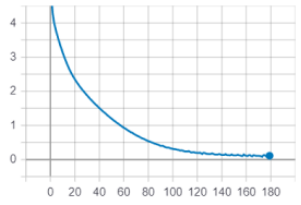
Visualization

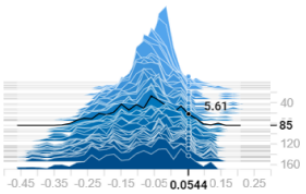
TRAINING HISTORY

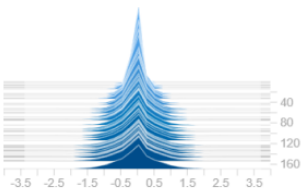
CONV. HISTOGRAMS

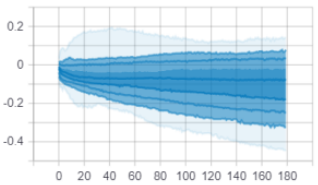
WEIGHT DISTRIBUTIONS

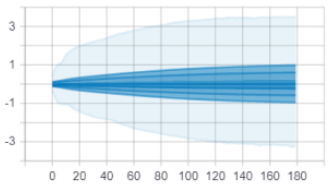












4

Design

System Flowchart