Mansee Agrawal 983635

Algorithms For Massive Data Project

FACE/COMIC RECOGNIZER – COMIC FACES

Declaration

"I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study"

Table of Contents

1. Introduction

In this project I am dealing with the comic or real face problem analysis stage of data AI development life cycle of the AMD project. In this project I am going to discuss about the problem identified in the problem statement, business problem statement, data collection, data preprocessing, train and test data splitting, model building, and deployment solution for the problem identified, proposed system, resource required.

For comic or real face Classification and identification large datasets and domain-specific features are used to best fit the data. In this project, I implemented train, and test state-of-the-art algorithms trained on domain general datasets for the task of identifying comic or real face.

2. Project Designing

Project design is an early phase of the project where a project's key features, structure, criteria for success, and major deliverables are all planned out. The point is to develop one or more designs which can be used to achieve the desired project goals.

2.1 Business Understanding

The entire cycle revolves around the business goal. What will you solve if you do not have a precise problem? It is extremely important to understand the business objective clearly because that will be your final goal of the analysis. After proper understanding only a person can set the specific goal of analysis that is in sync with the business objective. You need to know if the client wants to reduce credit loss, or if they want to predict the price of a commodity, etc.

2.2 Data Understanding

After business understanding, the next step is data understanding. This involves the collection of all the available data. Here you need to closely work with the business team as they are actually aware of what data is present, what data could be used for this business problem and other information. This step involves describing the data, their structure, their relevance, their data type. Explore the data using graphical plots. Basically, extracting any information that you can get about the data by just exploring the data.

2.3 Data Preparation

Next comes the data preparation stage. This includes steps like selecting the relevant data, integrating the data by merging the data sets, cleaning it, treating the missing values by either removing them or imputing them, treating erroneous data by removing them, also check for

outliers using box plots and handle them. Constructing new data, derive new features from existing ones. Format the data into the desired structure, remove unwanted columns and features. Data preparation is the most time consuming yet arguably the most important step in the entire life cycle. Your model will be as good as your data.

2.4 Exploratory Data Analysis

This step involves getting some idea about the solution and factors affecting it, before building the actual model. Distribution of data within different variables of a feature is explored graphically using bar-graphs, Relations between different features is captured through graphical representations like scatter plots and heat maps. Many other data visualization techniques are extensively used to explore every feature individually, and by combining them with other features

2.5 Data Modeling

Data modeling is the heart of data science. A model takes the prepared data as input and provides the desired output. This step includes choosing the appropriate type of model, whether the problem is a classification problem, or a regression problem or a clustering problem. After choosing the model family, amongst the various algorithm amongst that family, a person need to carefully choose the algorithms to implement and implement them. A person need to tune the hyperparameters of each model to achieve the desired performance. A person also need to make sure there is a correct balance between performance and generalizability. A person do not want the model to learn the data and perform poorly on new data.

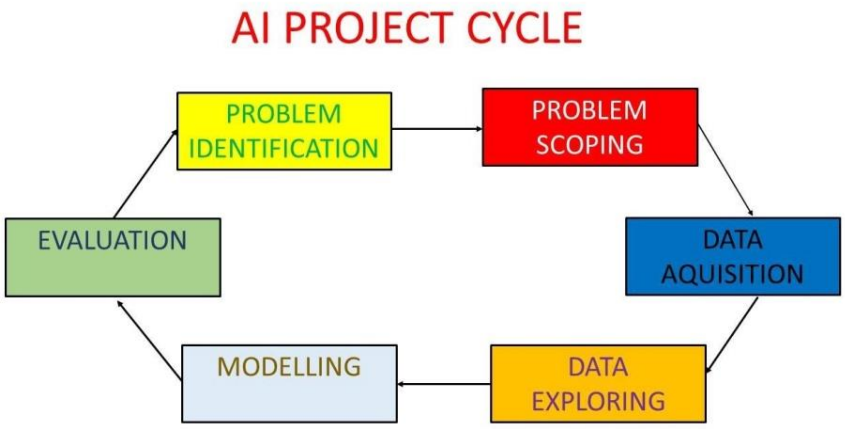
2.6 Model Evaluation

Here the model is evaluated for checking if it is ready to be deployed. The model is tested on an unseen data, evaluated on a carefully thought out set of evaluation metrics. A person also need to make sure that the model conforms to reality. If a person do not obtain a satisfactory result in the evaluation, a person must re-iterate the entire modeling process until the desired level of metrics is achieved. Any data science solution, a machine learning model, just like a human, should evolve, should be able to improve itself with new data, adapt to a new evaluation metric. A person can build multiple models for a certain phenomenon, but a lot of them may be imperfect. Model evaluation helps us choose and build a perfect model.

3. Lifecycle of AI project

Generally, the AI project consists of three main stages:

- Stage 1 – Project Planning and data collection
- Stage 2 – Design and training of the ML model.
- Stage 3 –Maintenance



Stage 1 – Project Planning and Data Collection

This is an initial stage, although very important and crucial as it explores the reasons why you decided to implement artificial intelligence solutions in your operations, as well as anticipating if the solution can be tangible and profitable.

You should think, analyze and evaluate if your problem can be solved with simpler solutions (like simple automation) or if it really requires more complex resources like Artificial Intelligence. In this project problem was to recognize if image is of real face or comic face.

Now after identifying problem you can focus on the solution. There are many solutions in the market that have worked for different businesses, but it is important to understand that your use case is unique and therefore requires a solution that is tailored to the needs of your business. So for solving problem in this project case I used Machine learning algorithm which is Deep learning Convolutional Neural Network (CNN).

From self driving cars to facial emotion recognition, data is at the core of all AI projects. Artificial Intelligence systems are capable of recognizing patterns and making decisions thanks to statistical models; for this to happen, data is required for the system to learn the correct patterns. However, when creating effective solutions, the challenge is not simply the availability of data, but rather the availability of large amounts of varied and high-quality data.Poor quality data not only prolongs projects, but also leads to more costly results, such as Machine Learning models not working properly and not delivering the desired results.Therefore, it is important that the data for your solution is not only of high quality but also relevant to the problem you are facing. My data for face recognition is of high quality because all the images are in HD quality. So there are 2 sources to collect data:

- ✓ Primary sources – Primary sources are sources that provide data that originates from your own company. You can acquire such data from tools like CRM or IoT devices.
- ✓ Secondary sources – Secondary sources refer to external sources that have relevant data of interest to you. Those can be Third-Party data providers or government publications.

Stage 2 – Design and training of the ML model.

Choosing the right ML model depends on a number of factors, such as the type of challenge your business is facing, the type of result you want to achieve, the size of your data, etc. Some of the types of ML models are:

- Binary classification model – As the name implies, ML models for binary classification problems predict a binary outcome. For example:
  Is this email spam or not (yes or no)?
  Is this app review written by a real person or not?
- Multiclass classification model – In this case, models predict an outcome into one of three or more classes. For example:
  Is a child holding a toy, a book, or a pen?
  Is this genre of music rock, jazz or hip hop?
- Regression model – Machine Learning models for regression problems predict the relationship between a single dependent variable and one or more independent variables. . For example:

  - ✓ **What will energy use be in California tomorrow?**
  - ✓ **How much product A will be sold this month?**

Training your model: In this important step, we will feed our data into our ML algorithms. This process gradually improves the ability of the models to produce the desired result (identify the object, predict the outcome, etc).

Like everything in life, practice makes better. This means that during the initial training process your model's output will have low accuracy, however this is normal and there is nothing to worry about, as with more and more training processes your model will improve. After training your model, as well as evaluating and testing its performance, it is time to move on to the next final stage.

Stage 3 – Maintenance

Finally, the last but not least stage, However, just because you've launched your AI solution live doesn't mean the project is done. As in the previous steps, an equally important part is monitoring, reviewing, and making sure that your solution continues to deliver the desired results.

4. Dataset

4.1 Data Download

The first step is to obtain the dataset for **Comic or Real face**. The direct download option through the command line is available on Kaggle's website. The command Kaggle datasets download requires the precise name of the item to be downloaded and the person's login credentials.

4.2 Data description

The dataset **Comic Faces** originates from a publicly accessible source, specifically Kaggle. This is a paired face to comic's dataset, which can be used to train pix2pix or similar networks.This dataset contains a lot of crappy nightmare-fuelish samples, which tend to be useful for full image to comic conversion, as the aim is to teach models to recognize whether it is a comic face or real face.

The dataset has total **20,000 images** which are either comic face or real face.
Sample of comic and real faces are shown below:

4.3 Data organization

The first step is to obtain the dataset Comic Faces. The direct download option through the command line is available on Kaggle's website. The command Kaggle datasets download requires the precise name of the item to be downloaded and the person's login credentials. Downloading Comic Faces Dataset from Kaggle. After we downloaded we have to manage our dataset with respective train and test folders for model training and Data organization with Default amount of 80% and 20%.

5. Kaggle API

The Kaggle API and CLI tool **provide easy ways to interact with Notebooks on Kaggle**. The commands available enable both searching for and downloading published Notebooks and their metadata as well as workflows for creating and running Notebooks using computational resources on Kaggle. In this project I used Kaggle for collecting data.

6. Data pre-processing

Data preprocessing, a component of data preparation, describes any type of processing performed on raw data to prepare it for another data processing procedure. It has traditionally been an important preliminary step for the data mining process. More recently, data preprocessing techniques have been adapted for training machine learning models and AI models and for running inferences against them.Data preprocessing transforms the data into a format that is more easily and effectively processed in data mining, machine learning and other data science tasks. The techniques are generally used at the earliest stages of the machine learning and AI development pipeline to ensure accurate results.
There are several Different tools and methods used for preprocessing data, including the following:

- Sampling, which selects a representative subset from a large population of data;
- Transformation, which manipulates raw data to produce a single input;
- DE noising, which removes noise from data;
- Imputation, which synthesizes statistically relevant data for missing values;
- Normalization, which organizes data for more efficient access; and
- Feature extraction, which pulls out a relevant feature subset that is significant in a particular context.

In this project I didn't do data cleaning and reduction because my data was already in a very good quality and data is relevant to ML tasks. Now for Data validation, the data is split into two sets. The first set is used to train a deep learning model. The second set is the testing data that is used to gauge the accuracy and robustness of the resulting model. This second step helps identify any problems in the hypothesis used in the cleaning and feature engineering of the data. Data is divided in 80% for training model and 20% for testing model.There are 20,000 images in data (10k comic faces and 10k real faces) from which 8,000 images of comic and real faces used for training and 2000 images for testing.

7. Train a PyTorch Classifier from Scratch

We will do the following steps in order:

1. **Load and normalize the Real vs Comic Custom Dataset for training and test datasets using TorchVision:**

   A lot of effort in solving any Deep Learning problem goes into preparing the dataset. Here we are using PyTorch provides TorchVision Image Folder to make out custom data loading easy and hopefully, to make your code more readable.

2. **Define a Convolutional Neural Network with Dense Layers**

   In Deep learning uses Artificial Neural Networks (models), which are computing systems that are composed of many layers of interconnected units. By passing data (Images, Text and more) through these interconnected units, a neural network is able to learn how to approximate the computations required to transform inputs into outputs.

   Our network will recognize real and comic face images. We will use a process built into PyTorch called convolution. Convolution adds each element of an image to its local neighbors, weighted by a kernel, or a small matrix, that helps us extract certain features (like edge detection, sharpness, blurriness, etc.) from the input image.

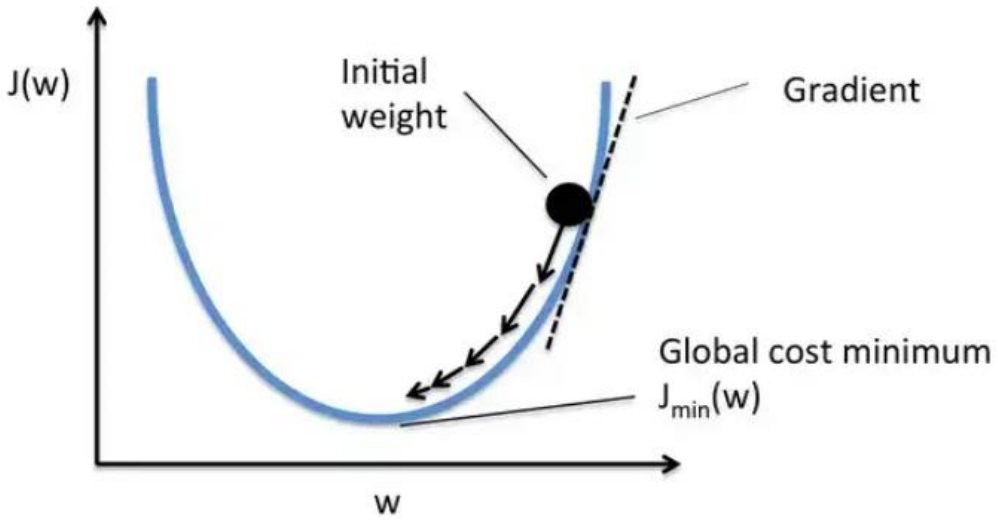
   In this Project I have designed neural network from scratch with multiple layers such as pooling layers, convolution layers, padding and finally fully connected layers.

```
Net(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2_1): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
  (conv2_2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1))
  (conv2_3): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1))
  (fc1): Linear(in_features=256, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=10, bias=True)
)
```

3. Define a loss function

Here I'm using Categorical Cross Entropy as loss function and Stochastic Gradient Descent (SGD as an Optimizer.

Error and Loss Function: In most learning networks, error is calculated as the difference between the actual output and the predicted output.



The function that is used to compute this error is known as Loss Function. Different loss functions will give different errors for the same prediction, and thus have a considerable effect on the performance of the model. One of the most widely used loss function is mean square error, which calculates the square of difference between actual value and predicted value. Different loss functions are used to deal with different type of tasks, i.e. regression and classification.

4. Train the network on the training data

This is when things start to get interesting. We simply have to loop over our data iterator, and feed the inputs to the network and optimize.

The CNN is a feed-forward network. During the training process, the network will process the input through all the layers, compute the loss to understand how far the predicted label of the image is falling from the correct one, and propagate the gradients back into the network to update the weights of the layers. By iterating over a huge dataset of inputs, the network will "learn" to set its weights to achieve the best results.

There are two ways to create neural networks in PyTorch using the Sequential method or using the class method. We'll use the class method to create our neural network since it gives more control over data flow. The format to create a neural network using the class method is as follows

Training Neural Network with Validation

The training step in PyTorch is almost identical almost every time you train it. But before implementing that let's learn about 2 modes of the model object:-

Training Mode: Set by model.train(), it tells your model that you are training the model. So layers like dropout etc. which behave differently while training and testing can behave accordingly.
Evaluation Mode: Set by model.eval(), it tells your model that you are testing the model.

Even though you don't need it here it's still better to know about them. Now that we have that clear let's understand the training steps:-

- Move data to GPU (Optional)
- Clear the gradients using optimizer.zero_grad()
- Make a forward pass
- Calculate the loss
- Perform a backward pass using loss.backward() to calculate the gradients
- Take optimizer step using optimizer.step() to update the weights

In our project code, we defined a neural network with the following architecture:-

Since we are using our custom dataset which is called real and comic face Classification images are of dimension 224*224 which have 50,176 pixels so when flatted it'll become the input to the neural network with 50,176 input nodes. We are using 3 channels with respective RBG.

Hidden Layer 1: 256 nodes
Max Pool Layer (2, 2)
Hidden Layer 2: 120 nodes
Hidden Layer 3: 84 nodes

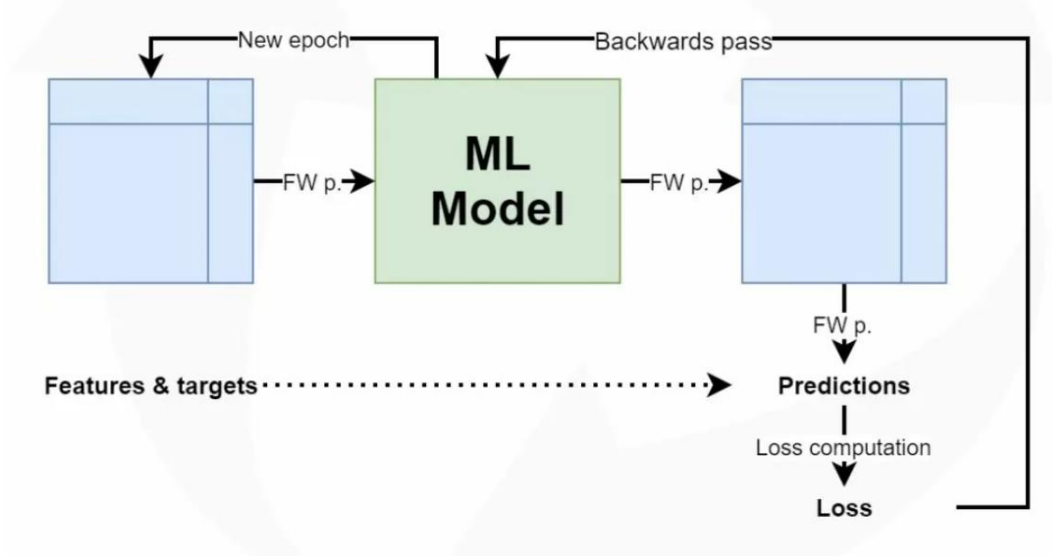
Here we are using 3 fully connected layers

Output Layer: 10 nodes, for 10 classes i.e. numbers 0-9

5. Test the network on the test data

At a high level, training a deep neural network involves two main steps: the first is the forward pass, and the second is the backwards pass and subsequent optimization.

When we start training a model, you'll initialize the weights and biases of the neurons pseudo randomly. During the first iteration, which is also called an epoch, all the data from your training set is fed through the model, generating predictions. This is called the forward pass. The predictions from this forward pass are compared with the actual targets for these training samples, which are called ground truth. The offset between the predictions and the targets is known as a loss value. At the beginning of a training process, loss values are relatively high.



Once the loss value is known, we perform the backwards pass. Here, we compute the contribution of the individual neurons to the error. Having computed this contribution, which is also known as a gradient, we can perform optimization with an optimizer such as gradient descent or Adam. Optimization slightly changes the weights into the opposite

direction of the gradients, and it likely makes the model better. We then start a new iteration, or epoch, and the process starts again.

Now, we can test the model with batch of images from our test set. For training model I used 35 epochs if you see below:

```
0 Epoch ============================== Completed...
[1,   200] loss: 1.939
1 Epoch ============================== Completed...
[2,   200] loss: 1.015
2 Epoch ============================== Completed...
[3,   200] loss: 0.541
3 Epoch ============================== Completed...
[4,   200] loss: 0.157
4 Epoch ============================== Completed...
[5,   200] loss: 0.070
5 Epoch ============================== Completed...
[6,   200] loss: 0.057
6 Epoch ============================== Completed...
[7,   200] loss: 0.038
7 Epoch ============================== Completed...
[8,   200] loss: 0.025
8 Epoch ============================== Completed...
[9,   200] loss: 0.021
9 Epoch ============================== Completed...
[10,   200] loss: 0.016
```

```
10 Epoch ============================== Completed...
[11,   200] loss: 0.011
11 Epoch ============================== Completed...
[12,   200] loss: 0.015
12 Epoch ============================== Completed...
[13,   200] loss: 0.009
13 Epoch ============================== Completed...
[14,   200] loss: 0.007
14 Epoch ============================== Completed...
[15,   200] loss: 0.008
15 Epoch ============================== Completed...
[16,   200] loss: 0.004
16 Epoch ============================== Completed...
[17,   200] loss: 0.005
17 Epoch ============================== Completed...
[18,   200] loss: 0.003
18 Epoch ============================== Completed...
[19,   200] loss: 0.003
19 Epoch ============================== Completed...
[20,   200] loss: 0.003
20 Epoch ============================== Completed...
```

```
25 Epoch ============================== Completed...
[26,   200] loss: 0.000
26 Epoch ============================== Completed...
[27,   200] loss: 0.000
27 Epoch ============================== Completed...
[28,   200] loss: 0.001
28 Epoch ============================== Completed...
[29,   200] loss: 0.000
29 Epoch ============================== Completed...
[30,   200] loss: 0.000
30 Epoch ============================== Completed...
[31,   200] loss: 0.000
31 Epoch ============================== Completed...
[32,   200] loss: 0.000
32 Epoch ============================== Completed...
[33,   200] loss: 0.000
33 Epoch ============================== Completed...
[34,   200] loss: 0.000
34 Epoch ============================== Completed...
[35,   200] loss: 0.000
Finished Training
```

At first I tried using more epochs but I got good results in less epochs so I just used 35 epochs for time saving and convenience.

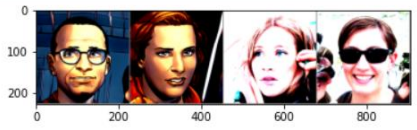
After using 35 epochs I checked how my model is working. Here as you can see below model is checking images correctly. I tried more than one time just to make sure that my model is really working good. I checked 5 instances from which one is showed below.

In [ ]:
```
import cv2
from torchvision.utils import save_image

dataiter = iter(testloader)
images, labels = next(dataiter)

# print images
imshow(torchvision.utils.make_grid(images[0:4]))
print('Classification Results: ', ' '.join(f'{classes[labels[j]]:5s}' for j in range(4)))
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```




```
GroundTruth:  comic comic real  real
```

8. Results

It is very clear that my model manage to reach to a good accuracy in both training and testing. My accuracy for comic face is 99.8% and for real face is 99.8%.

```
# Accuracy form each class
for classname, correct_count in correct_pred.items():
    accuracy = 100 * float(correct_count) / total_pred[classname]
    print(f'Accuracy for class: {classname:5s} is {accuracy:.1f} %')
```

```
Accuracy for class: comic is 99.8 %
Accuracy for class: real  is 99.8 %
```

9. Conclusion

Face recognition technology has come a long way in the last twenty years. Today, machines are able to automatically verify identity information for secure transactions, for surveillance and security tasks, and for access control to buildings etc. These applications usually work in controlled environments and recognition algorithms can take advantage of the environmental constraints to obtain high recognition accuracy. However, next generation face recognition systems are going to have widespread application in smart environments -- where computers and machines are more like helpful assistants.

The implementations and results demonstrate that deep learning techniques have great Performance in Face detection. We can even use Transfer learning model to minimize time and also for the better performance of the model because you can get quite good results. I also tried using transfer learning model and I got really accuracy for both training and testing model.
But building model from scratch also gives good accuracy but it take little more time comparatively.
I would suggest that you should have GPU machine if you want to run code for deep learning for a large number of datasets because it takes comparatively lesser time whereas if you don't have GPU machine then it will take lot of time to run epochs because of large dataset.
Finally, there is still no robust face detection and recognition technique for unconstraint real-world applications.

For future work on this,we can convert this model to develop for mobile application just similar to snapchat. We can also try to convert this model into age detection application or gender detection application by changing the data accordingly.