**APS360 Team 53 Progress Report**

**Shixuan (Shawn) Huang**

Student# 1006964171

shixuan.huang@mail.utoronto.ca

**Qingran Chen**

Student# 1006724751

qingran.chen@mail.utoronto.ca

**Jiahe Lu**

Student# 1007213604

joeyjiahe.lu@mail.utoronto.ca

**Yunxiang Zhang**

Student# 1005985089

vincentyunxiang.zhang@mail.utoronto.ca

# **1.0 Brief Project Description**

Provide a brief description of the motivations behind your project, the goal of your project, why it is interesting or important, and why deep learning is a reasonable approach.

◦ Someone unfamiliar with the project should be able to quickly establish the project goals, why the project is interesting and/or useful, and why machine learning is an appropriate tool for the task.

◦ Provide supporting images and diagrams to minimize the required reading.

◦ Visual elements are professional, concise and easy to read.

◦ Make sure you show what is the input and what is the output of your model and justify why deep learning is a good approach to generate such outputs from your inputs.

## *1.1 Motivation*

The motivation behind this project arises from the escalating prevalence of Deepfake technology, which poses significant threats to the integrity of digital media. Deepfakes, sophisticated digital forgeries that manipulate audio and visual content to create deceptive representations of individuals—undermine trust in media and have far-reaching implications in politics, security, and personal privacy.

The goal of this project is to develop a robust detection system capable of distinguishing between genuine and manipulated content with high accuracy. To achieve this, the team has selected a comprehensive dataset from Kaggle [], contributed by Tushar Padhy, encompassing over 140,000 portrait images of varying authenticity. This dataset serves as a foundational element for training and evaluating the model, ensuring a diverse representation of real and fake examples. The choice of deep learning, specifically Convolutional Neural Networks (CNN), stems from its proven effectiveness in image recognition tasks. Deep learning's ability to process and learn from large volumes of data, identify complex patterns, and adapt to evolving challenges makes it an ideal approach for addressing the sophisticated nature of deepfake generation and detection.

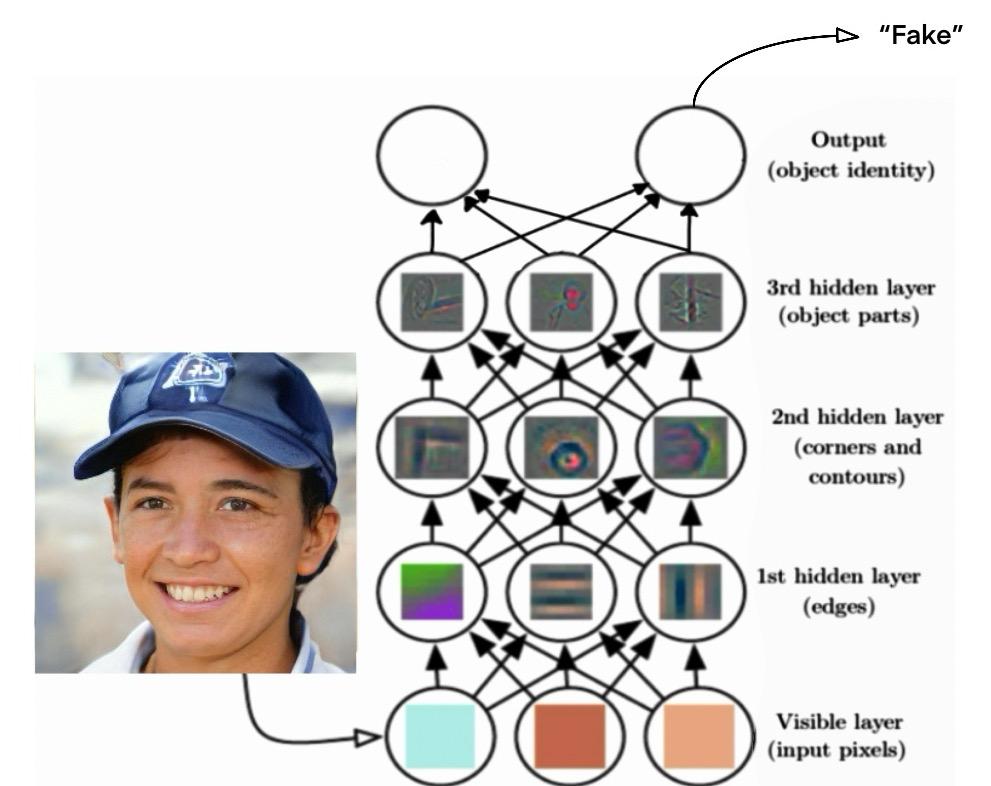
## *1.2 Model Architecture and Applied Methodology*

The project's workflow is designed around the development, evaluation, and refinement of a CNN-based deepfake detection model. The project flow begins with the conduction of a visual inspection with random sampling on the pre-divided dataset across all segments—training, validation, and testing—to identify and remove duplicate images from individuals and across categories to ensure a comprehensive evaluation framework. The **baseline model** serves as the initial point of reference, employing a simple CNN architecture to establish a performance benchmark. This model comprises convolutional layers for feature extraction, pooling layers for dimensionality reduction, and fully connected layers for classification. The **primary model**, intended to surpass the baseline, incorporates advanced techniques such as deeper convolutional layers, dropout for regularization, and potentially alternative architectures like residual networks (ResNets) or transformers to enhance detection capabilities.

Data separation plays a crucial role in model development and evaluation. The training set is used to teach the model to distinguish between real and fake images, the validation set aids in tuning hyperparameters and preventing overfitting, and the testing set provides an unbiased assessment of the model's performance. This structured approach facilitates iterative refinement, enabling the team to systematically improve the model's accuracy and reliability in identifying deepfakes.

## *1.3 Idea Illustration of Model Input and Output*

The model is trained using input images resized to uniform dimensions, such as 256x256 pixels, each labelled as "Real" or "Fake". For instance shown in Figure 1, an input is a 256x256 pixel image of a person's face, with the model outputting a binary classification: 1 (Real) or 0 (Fake), indicating the image's authenticity.



**Figure 1. Sample Input and Output Illustration**

The team employs a systematic process to tune the model's parameters, including the learning rate, number of convolutional layers, and size of the filters, aiming to optimize performance. Through training and validation, the model learns to recognize and generalize from the distinguishing features of real and fake images. The desired accuracy is set at a high threshold to ensure the model's effectiveness in practical applications. Achieving this level of accuracy requires meticulous parameter tuning and extensive evaluation, underscoring the project's commitment to developing a reliable and robust deepfake detection system.

# **2.0 Individual Contributions and Responsibilities**

Provide a summary of how your team is working together. Describe the project management software that you are using to communicate with each other, track progress and results, track updates to shared code, etc. You should provide details on what each person is responsible for and has accomplished so far, along with an updated list of tasks and deadlines for each member.

◦ It is expected that you will provide a great deal of detail on what the expectations are for each team member and what they have completed to date.

◦ The progress of the team is well summarized, and it should be easy to determine if the team is on track, falling behind, or unlikely to complete the project.

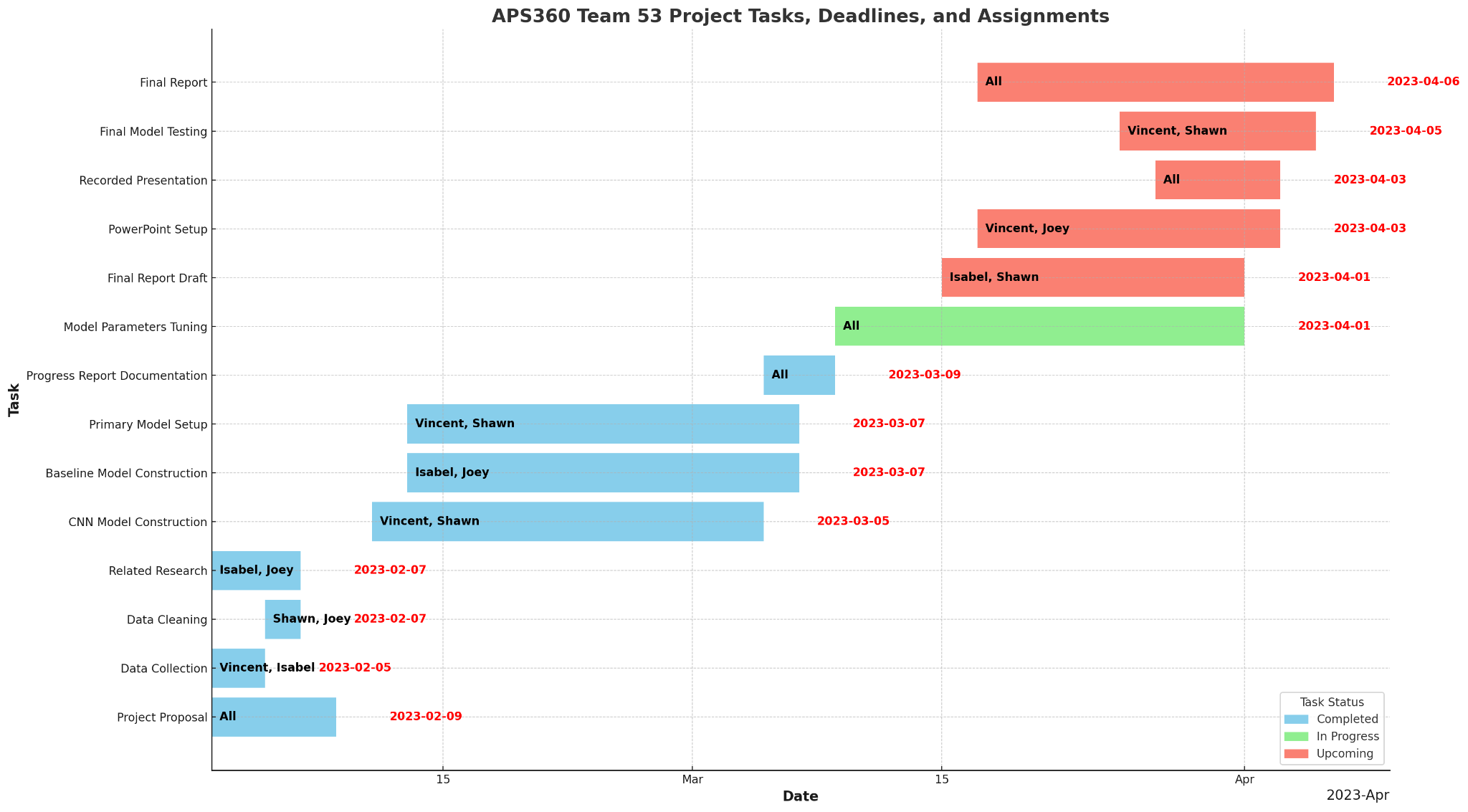
◦ The contributions and quality of communication are clearly and concisely documented for each team member. If a member has not been contributing equally to the team or failing to meet the

deliverables

◦ You are expected to provide redundancies for some special (sensitive) tasks that a member is responsible to do to address the any risks that may prevent you from completing the project, or that plans for how you can obtain help from your other team members.

◦ The summary is professionally written, concise and easy to read.

The team has made significant progress in this project to detect Deepfakes using Convolutional Neural Networks (CNN), with a structured and collaborative approach to task assignment and execution. The team communication is performed at Messager and the project progress is tracked through an updated Gantt chart **(Figure 2)**, which outlines the tasks, deadlines, and assignments, ensuring that the project's milestones are met efficiently.



**Figure 2. Team Tasks Gantt Chart Screenshot**

Vincent and Isabel efficiently handled data collection from February 2nd to 5th, assembling a dataset of over 300,000 images for the project's needs. Concurrently, undertook the data cleaning process from February 5th to 7th, focusing on duplicate removal and quality assurance. In parallel, a literature review conducted by Isabel and Joey from February 2nd to 7th yielded critical insights into deepfake detection methodologies and Convolutional Neural Networks (CNNs), thereby informing the project's strategic direction.

Following these foundational tasks, Vincent and Shawn focused on the CNN model's construction from February 11th to March 5th and developed an optimized architecture for Deepfake detection. By March 7th, the collective team had successfully established a baseline model and completed the advancement of the primary model. Additionally, a comprehensive progress report was compiled by March 9th, providing a detailed account of the methodology employed and the preliminary results obtained.

Currently, the team is engaged in the model parameters tuning phase, which began on March 9th and is set to be completed by April 1st. This critical phase involves all team members. The upcoming tasks include drafting the final project report, setting up the PowerPoint presentation, recording the final presentation, conducting the final model testing, and compiling the final report, with deadlines ranging from April 1st to April 6th.

To mitigate risks and ensure project completion, tasks have been designed as assigned to two members as a group. Such as data cleaning and model testing. This strategy ensures that, should any team member encounter unforeseen circumstances, the project can continue without significant delays. The team has regularly updated each other on progress and challenges through weekly meetings and shared documentation platforms.

To date, the team is on track to completing the project within the established timeline. The contributions of each team member have been vital to the project's progress, with tasks completed efficiently and effectively.

**3.0 Data Processing**:

You can describe the data that you have collected and cleaned. Be clear andspecific when describing what you have done, to the point that someone could reproduce your work. If possible, show some statistics about your cleaned data (e.g., number of examples in each class), and at least one example of a cleaned training sample. You should also mention how you plan to obtain new data for final testing of the model. We prefer you collect some new data yourself or if that is not possible, use an entirely different dataset which was not used anywhere before.

◦ Clearly describes sources of data, and the steps you took to clean and format your data. Statistics and data examples are well chosen and show that you have completed data processing. Data cleaning descriptions are clear enough to be reproduced by a classmate.

◦ Visual elements are professional, concise and easy to read.

◦ Provides a plan for testing on never before seen data.

◦ Describe any challenges you may have faced with the task.

**Data overview:**

This dataset contributed by Tushar Padhy, contains a significant assembly of more than 140,000 portrait images, specifically curated to train deep learning models within the scope of image authentication endeavours. A notable feature of this dataset is its balanced representation in terms of gender, along with comprehensive coverage across a diverse range of ages and ethnicities.

**Data cleaning:**

Duplicate Elimination: The project team employed a hash function technique to ascertain and eliminate precisely identical duplicates, resulting in a 2% reduction in the overall dataset size.

Image Normalization and Resizing: To ensure uniformity across the dataset and to minimize computational demands during the training phase, all images were resized to a standardized resolution of 224x224 pixels. Furthermore, the pixel values were normalized to a range between 0 and 1.

Label Accuracy Assessment: A random subset of 20 images was subjected to a manual inspection by our team to verify the correctness of their labels. Any instances of mislabeling identified during this review process were duly corrected.



Figure 1 : 20 samples after normalization

**Data categorizing:**

The process of preparing and categorizing image data into train, validation by Python and test groups for a binary classification task involves a systematic approach to ensure efficient model training and evaluation. This process involves using TensorFlow’s Data Generator for loading and preprocessing image data for training a neural network. The explanation for each step in the categorizing process[Figure 2 python code]:

1. Directory Structure: Dataset was loaded from Kaggle using API and zas unzipped into Google Drive. Images are thoroughly organized into distinct 3 directories for training, validation, and test datasets with the cleaning of hash functions. Within each of these, images are further classified into subdirectories named 'Fake' and 'Real', corresponding to their labels(0 and 1). The labelled data will be automatically processed with a data generator during the loading process.

2. Training Data Processing: To enhance model robustness and prevent overfitting, the training images are shuffled. This randomization ensures the model encounters a diverse mix of data points across epochs, mitigating the risk of learning from the sequence rather than the features of the images.

For the training data, an ImageDataGenerator is created with several data augmentation parameters:

* Rescaling: Each pixel value is rescaled by 1./255 to convert pixel values from the range [0, 255] to [0, 1] for neural network processing.
* Rotation: Images are randomly rotated within 15 degrees.
* Width and Height Shift: Images are randomly shifted horizontally and vertically by 10% of the total width and height.
* Shear: A random shearing transformation is applied.
* Zoom: Random zooming of images within 20%.
* Horizontal Flip: Images are randomly flipped horizontally.
* Fill Mode: Specifies how the input image is filled after transformations. nearest fills with the nearest pixels.

This generator then creates a training data loader using flow\_from\_directory, which automatically labels images based on the directory structure, resizes images to 227x227 pixels(255 x 255 for the primary model), and specifies the batch size.

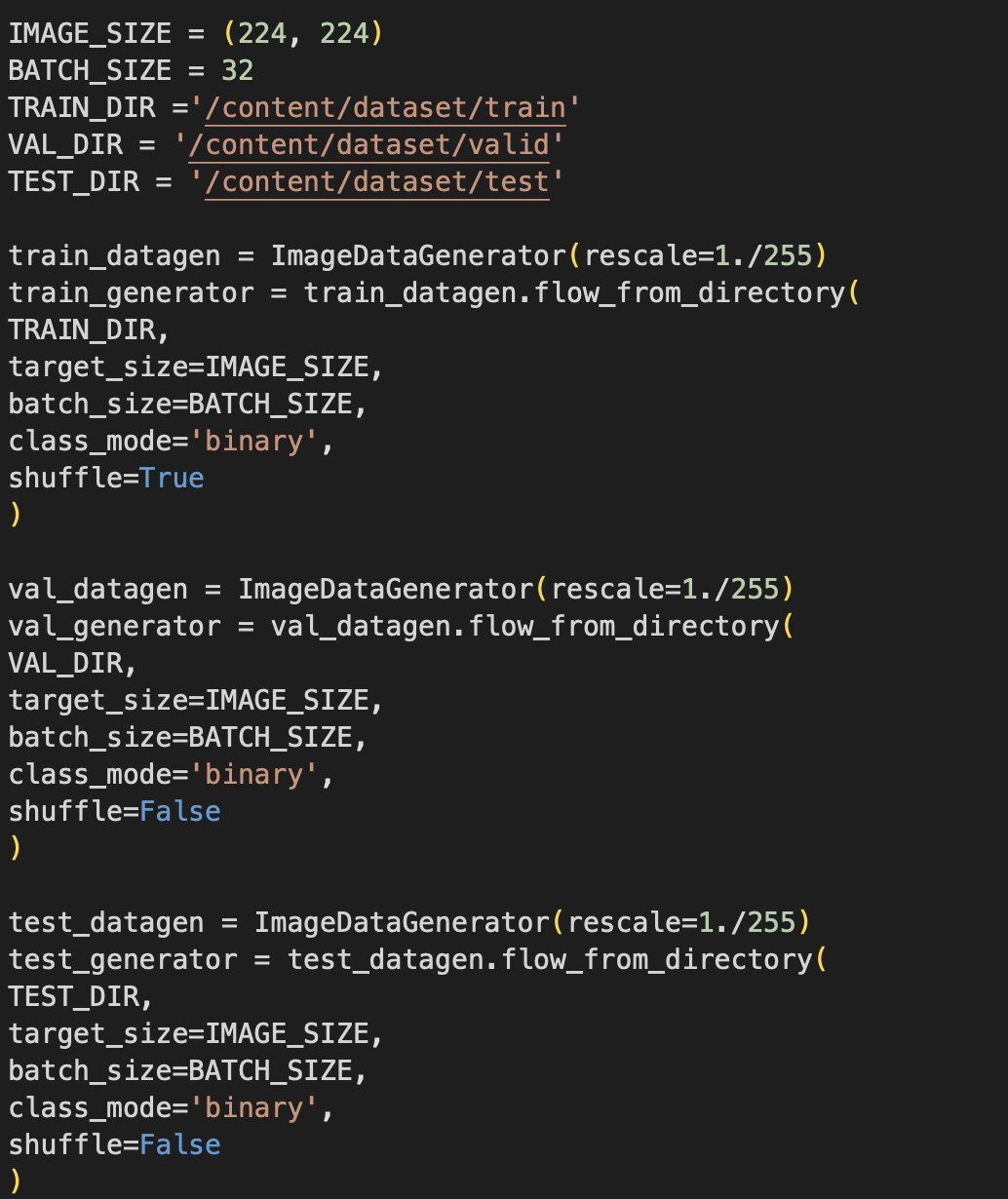
3. Validation and Test Data Processing: A separate ImageDataGenerator is used for validation and test data, which only rescales the pixel values without applying data augmentation. This is important to evaluate the model on unmodified images. Data loaders for validation and test datasets are created similarly to the training data loader, using flow\_from\_directory with the same target size and batch size. The key difference is the lack of data augmentation to keep the evaluation and testing conditions consistent and realistic.

4. Data Categorization: Leveraging the `class\_mode='binary'` option, the data loading mechanism is informed that the task at hand is binary classification. This setting automates the process of assigning binary labels (0 or 1) to the images based on their subdirectory ('Fake' or 'Real'), simplifying the preparation of labels for model training and evaluation.

5. Batch Processing: Images are processed in batches of 256, optimizing the use of computational resources. Batch processing helps in utilizing computational resources efficiently and can impact the training dynamics and model performance, strikes a balance between computational efficiency and the gradient descent's need for diverse data points to navigate the solution space effectively

**Data implemented in model testing:**

After processing the datasets with the dataset\_train\_processing() function, which prepares the data generators and counts the samples for training, validation, and testing sets, data generators were proceeded to train the model using model.fit() function from TensorFlow. During training, the model will use the training generator to receive batches of data, with the number of steps per epoch determined by dividing the total number of training samples by the batch size. Validation is performed at the end of each epoch using the validation generator and the total number of validation samples divided by the batch size to determine the validation steps.



**Baseline:**

The Baseline chosen for the group is AlexNet. AlexNet consists of 5 convolution layers, 3 max-pooling layers, 2 Normalized layers, 2 fully connected layers and 1 SoftMax layer~\citep{Bhagwat2021}. The structure of the AlexNet can be refered to Figure~\ref{Alex}

**The Choice of AlexNet:**

AlexNet is a large, deep convolutional neural network to classify the 1.2 million high-resolution images and won the ImageNet LSVRC-2010 contest into the 1000 different classes. The Convolutional neural networks (CNNs) can make make strong and mostly correct assumptions about the nature of images (namely, stationary of statistics and locality of pixel dependencies). As an beginning model of DeepLearning, it's a great start point for the project to be referenced as a baseline. In addition, as pointed out in Dr.Krizhevsky's paper, in 2012 the AlexNet's size was limited by GPU performance~\citep{KrizhevskySutskeverHinton2012}, the primary model of this projects will build an CNN model that imitates the GoogleLeNet with residual learning and building a deeper layer model for this binary classification task.

**Challenges Faced:**

In order to address the challenge, the project adopted the keras and tensorflow instead for the baseline model. TensorFlow enhances efficiency with its streamlined data loading(data generator)and preprocessing features, essential for handling our large datasets effectively. Moreover, keras grants access to pre-trained models such as AlexNet, offering a solid foundation thanks to its pre-configured processing and optimized architecture. This decision enables us to utilize these sophisticated capabilities effortlessly, smoother our project's development. ~\citep{InsigniteAlexnetDogvsCat}

**Quantitative and Qualitative Result Achieved:**

Due to the limitation of time, the baseline model only performs 3 epoch with 256 batch size, 0.001 learning rate using adam optimizer. The result was demonstrated in Figure~\ref{Epoch}

The validation and train accuracy was also shown in Figure~\ref{Val}, the blue line represent train result, red line represent the validation result, the upper graph shows the train and validation loss, the lower graph shows the accuracy. The training loss is decreasing with validation loss keeps stationary. The gap between training and validation accuracy where the training accuracy is significantly higher than the validation accuracy also points towards over-fitting. These observations suggest that the model may be over-fitting to the training data. Further adjustment to the baseline model will be conducted including data augmentation, tuning hyper-meters and early stop strategy.

# 

Figure 2: Python code for data categorizing

**Challenges in data process:**

### 1. Memory Challenges:

* Large In-Memory Requirements: Large datasets can exceed the memory capacity of your hardware, leading to crashes or severely degraded performance.
* Efficient Data Loading: Strategies such as batch processing, data generators, or utilizing databases to load data on demand are essential to manage memory usage effectively.

2. Computational Challenges:

* Processing Power: Large datasets require significant computational power for processing and model training, which can lead to long training times.
* Parallel Processing: Utilizing GPUs or distributed computing can help, but also introduces complexity in code and infrastructure setup.

### 3. Data Quality and Consistency

* Inconsistent Data Formats: Diverse sources of data can lead to inconsistencies in format or quality, necessitating extensive preprocessing and normalization.
* Missing or Noisy Data: Handling missing values, errors, or noise in the data can be challenging, requiring techniques like imputation, filtering, or augmentation.

– **Baseline Model**:

You can provide a diagram to describe the baseline model that you have tested and how it was/will be compared with your primary neural network model.

◦ A reasonable choice of baseline for the problem being solved.

◦ ◦ The best results achieved with your model in terms of quantitative (i.e., accuracy, error, loss) and qualitative (i.e., identify something interesting about how your model performs on select samples or class of samples).

◦ Describe any challenges you may have faced with the task.

– **Primary Model:** You can provide a diagram to describe the best model architecture that you have so far. This description should be more detailed than in your initial proposal. You should provide a rough idea of how complex your model is (e.g., number of layers, number of parameters, etc.),

and all the details necessary so that someone can reproduce a model like yours that will perform similarly. You should also provide at least one quantitative and qualitative result. These results could be learning curves, or results showing the performance of the model on selected samples of data. The focus here is on assessing the feasibility of your model to achieve the project objectives.

◦ The choice of architecture makes sense for the problem.

◦ The architecture implementation is based on neural networks.

◦ Visual elements are professional, concise and easy to read.

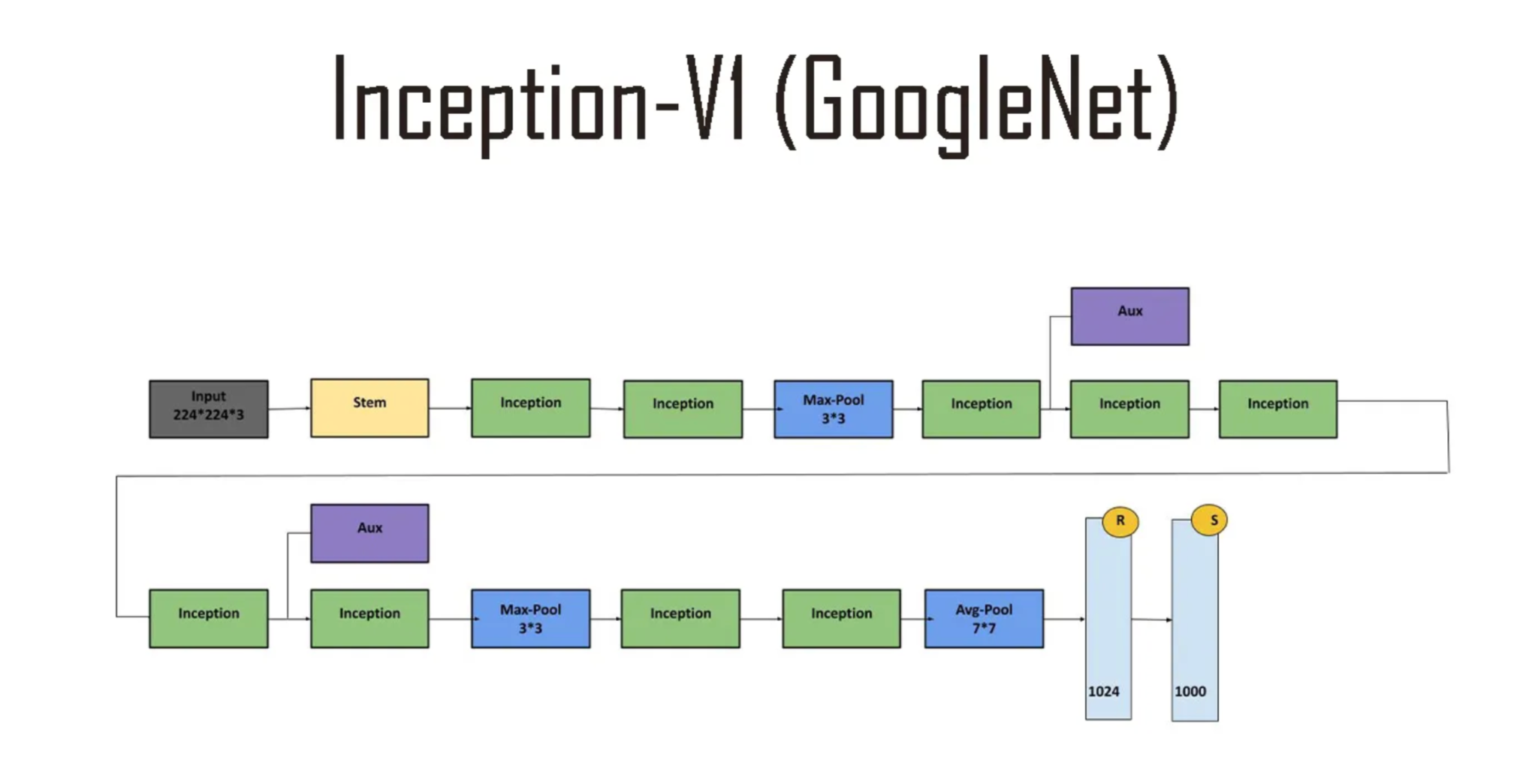
◦ The best results achieved with your model in terms of quantitative (i.e., accuracy, error, loss) and qualitative (i.e., identify something interesting about how your model performs on select samples or class of samples).

◦ Describe any challenges you may have faced with the task.

# 5. Primary Model:

The primary model chosen for the project is GoogLeNet.

GoogLeNet is built from the ideas of basic Convolutional Neural Network, and it consists of 22 convolutional layers and multiple inception blocks with filter sizes from 1\*1 to 5\*5. The variation of the filtering sizes allows given inputs to be passed through different convolutional layers with different sizes and the max poolings are in parallel. Therefore, the network could capture information from various scales and complexities and be able to decide how to filter the size of the input images best.



5.1 Choice of GoogLeNet:

According to the insights gained from the lecture and subsequent research, it has been identified that GoogLeNet offers superior training accuracy and parameter efficiency relative to AlexNet. Specifically, GoogLeNet is comprised of approximately 4 million parameters, in stark contrast to AlexNet's 60 million parameters. Despite this significant reduction in parameter count, GoogLeNet demonstrates enhanced performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), achieving a notable error rate of 6.67% in the classification of 1.2 million high-resolution images across 1000 categories, thereby surpassing AlexNet's accuracy[].

5.2 Challenges Faced

The GoogLeNet was initially designed for multi-classification tasks. This means the activation function used is “softmax” and the number of neurons in the output layer is 1000. To adapt the neural network to the binary classification class, we must adjust the last output layer to 1 neuron and the activation function to “sigmoid.”

The large number of layers means the large number of hyperparameters needs to be tuned for higher performance. According to the structure of the neural network, as the image goes in from the input layer, it passes through many inception blocks that creates different filters with different convolutional layers and pooling layers. Therefore, tuning becomes more challenging because a tuning on one hyperparameter can affect other layers and the output.

5.3 Qualitative and Quantitative Result Achieved

Regrettably, the primary model did not achieve the expected results. In the context of the ImageNet competition, GoogLeNet reported an error rate of 6.67% across tasks involving 1000 multi-class classifications. The objective for the model, when applied to a binary classification task, was to secure an error rate below 10%. Despite these aspirations, during the initial training phase covering the first five epochs, the model's accuracy remained stagnant at 0.5, with no signs of improvement. However, there was a notable decrease in loss, descending from a significantly high value of approximately 90 to around 10. This reduction in loss indicates that the network is learning, as evidenced by the convergence of the output towards the target. Nevertheless, the stagnation of accuracy levels implies that the model's performance has not been enhanced through the training process.

Several factors could be contributing to this outcome:

1. Insufficient Number of Epochs: Likely, given that the loss remained above 1 at the conclusion of the last epoch, indicating ongoing learning but an inability to accurately classify the images.
2. Data Loading Issues: Unlikely, considering the data was carefully normalized and split.
3. Training Network Concerns: Unlikely, given the utilization of Keras's built-in model.fit() function.
4. GoogLeNet Architecture Compatibility: Plausible, especially since adaptations were made to the structure to accommodate a binary classification task, diverging from its original multi-classification design.
5. Hyperparameter Optimization: Likely a contributing factor, as modifications were limited to the output layer's activation function and neuron count, while other hyperparameters remained unadjusted.

Exploring these potential issues could improve the network's learning capability and, consequently, its classification accuracy.