**APS360 Team 53 Final 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 Introduction**

The impetus for this project is the rapidly growing prevalence of deepfake technology, which presents a significant challenge to the integrity of digital media. Deepfakes, which are advanced digital forgeries manipulating audio and visual content to create convincingly deceptive portrayals of individuals, erode trust in media. They carry profound consequences across various sectors, including politics, security, and personal privacy, by fostering misinformation and potentially endangering individuals' reputations.

Our objective is to develop a highly accurate detection system capable of differentiating genuine content from manipulated ones. To this end, our team has procured a comprehensive dataset from Kaggle, curated by Tushar Padhy in 2024, which includes over 140,000 portrait images featuring a mix of authentic and altered content. This dataset is pivotal for training and fine-tuning our model, ensuring it has a broad spectrum of examples to learn from and can thereby recognize a wide range of deepfake techniques.

The methodology underpinning our approach is the deployment of Convolutional Neural Networks (CNN), a choice motivated by their demonstrated success in image analysis tasks. CNNs are a cornerstone of deep learning technologies, renowned for their proficiency in processing and learning from extensive datasets, discerning intricate patterns, and their adaptability to new, sophisticated challenges. This makes them exceptionally suited to tackle the complex and evolving landscape of deepfake generation and detection, aiming for a future where digital trust can be restored and maintained.

# **2.0 Illustration/Figure**

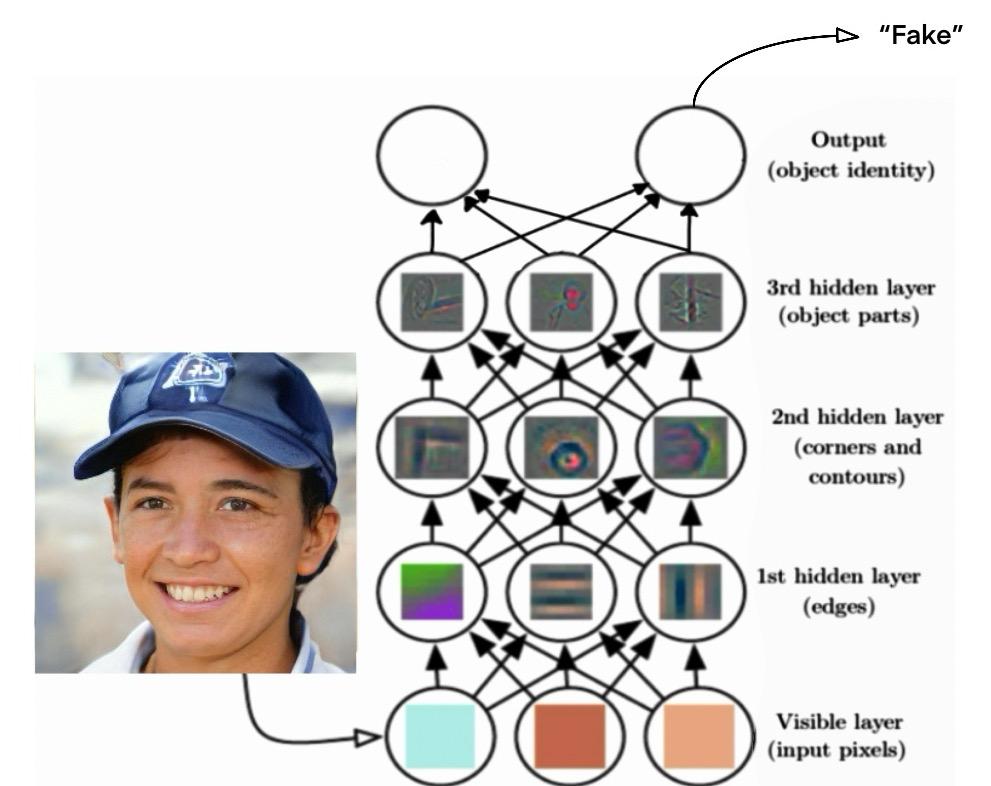
To effectively communicate the central mechanism of our deepfake detection project, we propose including a schematic that outlines the key stages of our model's operation. This figure should serve as an at-a-glance overview for readers, making the complex workings of our system immediately understandable, even to those with minimal technical background.

The diagram will depict the following key components and flow:

1. Input Stage: An illustration showing that input images are standardized to a uniform size of 256x256 pixels. This normalization is crucial for ensuring consistent analysis across varying inputs.

2. Processing Stage: A representation of the image being processed by our Convolutional Neural Network (CNN) model. This could be symbolized by layers or nodes that highlight the deep learning aspects of the system.

3. Output Stage: The final part of the diagram will show the model's output, which is a binary classification indicating the authenticity of the image: 1 (Real) or 0 (Fake). This demonstrates the model's capability to discern between genuine and manipulated content.



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

For illustrative purposes, the figure (let’s refer to it as Figure 1) will include an example: a 256x256 pixel image of a person's face, clearly labeled as "Input." This flows into a stylized depiction of the CNN model, eventually leading to the output labeled either "1 (Real)" or "0 (Fake)" based on the model’s assessment.

To ensure clarity and accessibility, the figure could be created using a digital drawing tool for neatness and legibility, though hand-drawn diagrams are equally acceptable provided they are clear. The use of PowerPoint or similar software is recommended for those seeking a balance between professionalism and ease of creation. Remember, the effectiveness of the figure in conveying the project’s core idea and architecture is paramount, not the artistic skill or the medium used.

# **3.0 Background & Related Work**

A description of 5 related work in the field, to provide reader a sense of what has already been done in this area, e.g. papers or existing products/software that do a related thing.

Briefly describes 5 prior work related to your project to put your project into context. Your descriptions need not be complete, but should contain important work.

The team reviews five pivotal works and initiatives in the realm of deepfake detection, offering insight into the advancements and methodologies employed to combat digital deception:

**Kaggle Deepfake Detection Challenge (2020):** Spearheaded by major tech entities such as AWS, Facebook, and Microsoft, alongside academic partnerships, the DFDC initiative emerged as a comprehensive effort to curtail deepfake content through collaborative competition. Utilizing a log loss function for model evaluation, it provided participants with an expansive dataset, serving as a critical benchmark for developing deepfake detection algorithms.

**Sentinel AI**: This platform represents a significant advancement in AI-driven security, aiding democratic institutions and businesses in mitigating the threats posed by deepfakes. Employed by top European organizations, Sentinel AI leverages neural network classifiers to refine its detection capabilities, setting a precedent for future technological solutions in this domain.

**Human vs. Machine Deepfake Detection (2021, Matthew Groh)**: Investigating the efficacy of human intuition against machine precision, this study revealed that machines generally surpass individual human accuracy in identifying deepfakes, achieving an 80% success rate. Notably, when AI predictions aided human participants, their detection accuracy improved, underscoring the potential synergy between human insights and artificial intelligence.

**Deep Learning Approaches to Deepfake Detection (2021, Abdulqader M. Almars)**: This survey paper provides a succinct overview of deepfake creation and detection, examining the utilization of CNNs, RNNs, and hybrid models. It emphasizes the significance of diverse and robust datasets in training detection systems, highlighting the foundational role of data variety in enhancing model reliability.

**CNN vs. Transformers in Deepfake Detection (2023, Vrizlynn L.L.)**: This comparative study delves into the performance of eight deep learning architectures, including CNNs and Transformers, across various datasets. Findings indicate that CNNs perform better in uniform dataset evaluations, whereas Transformers excel in cross-dataset scenarios. The analysis sheds light on the nuanced strengths of different architectures, suggesting a tailored approach to model selection based on the specific challenges of deepfake detection.

# **4.0 Data Processing**

Describe the data that you have collected and if you have preprocessed it in any way. Be clear and specific when describing what you’ve done, so that a classmate can reproduce your work. Show some statistics and examples of your data. The extent of data processing will vary from project to project and you will be graded accordingly.

Clearly describes sources of data, and the steps you took to clean and format your data. Statistics and data example are well-chosen, and gives readers a “feel” for your data.

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.

**4.1 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 2: 20 samples after normalization

**4.2 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

**4.3 Data implemented in model testing:**

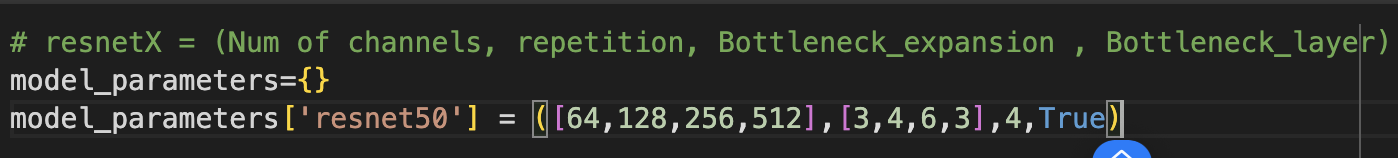
After processing the datasets with the dataset\_train\_processing() function, which prepares the data generators and counts the samples for training, validation, and testing sets, data generators were proceeded to train the model using model.fit() function from TensorFlow. During training, the model 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.

# **5.0 Architecture**

A description of the final neural network model architecture. Do not describe all the intermediate models that you have tried. Instead, present the model (or models) whose quantitative results you will show. These should be your most interesting models. Be as specific as you can while being concise. Readers should be able to reproduce a model similar enough to yours and obtain a similar performance.

◦ 4/4 Clear and concise description of your model architecture, so that a classmate can reproduce a model similar to yours that will perform similarly.

Our final neural network model architecture employs a Residulal Network (ResNet) architecture optimized for deepfake detection. This model was chosen for its ability to process and analyze complex image data effectively, distinguishing genuine from manipulated media with high accuracy.

**Model parameter**:

The model parameters in the configuration represents number of channel, repetition, bottleneck expansion, and bottleneck layer respectively.

The number of channel indicate the number of feature maps at each stage of the network[1], is represented by the array [64, 128, 256, 512]. The model has 4 stages, the first stage has 64 channels, and it doubles every instance when it reaches the next stage. These are the hyperparameter we can tune later in the process.

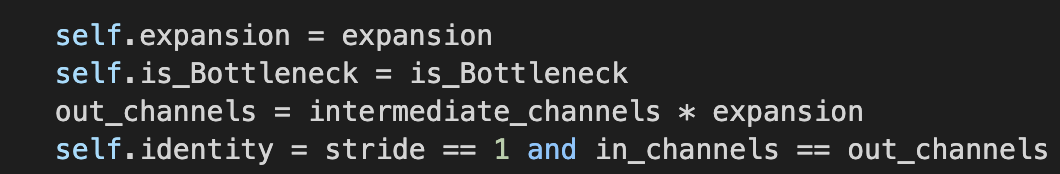
Repetition is also a hyperparameter we can tune in the model. It represents the number of times the block will repeat in the each stage[1]. The repetition allows the model to learn more complex feature without adding architectural complexity, however the time and computational cost still apply.

Resnet50 applies the bottleneck block, where as an input image pass through the first 1\*1 convolutional layer that reduce the number of channels, and then as it pass through the next convolutional layer, the input get expanded by a factor of 4 after it is passed to the intermediary 3\*3 convolutional layer and final 1\*1 convolutional layer[1], the purpose is to increase efficiency.

The bottleneck layer is always used when using resnet50 and up. The bottleneck layer increases the network depth for enchanting learning capacity while managing the computational efficiency[1].

**Bottleneck Block**

**Initialization:**

****

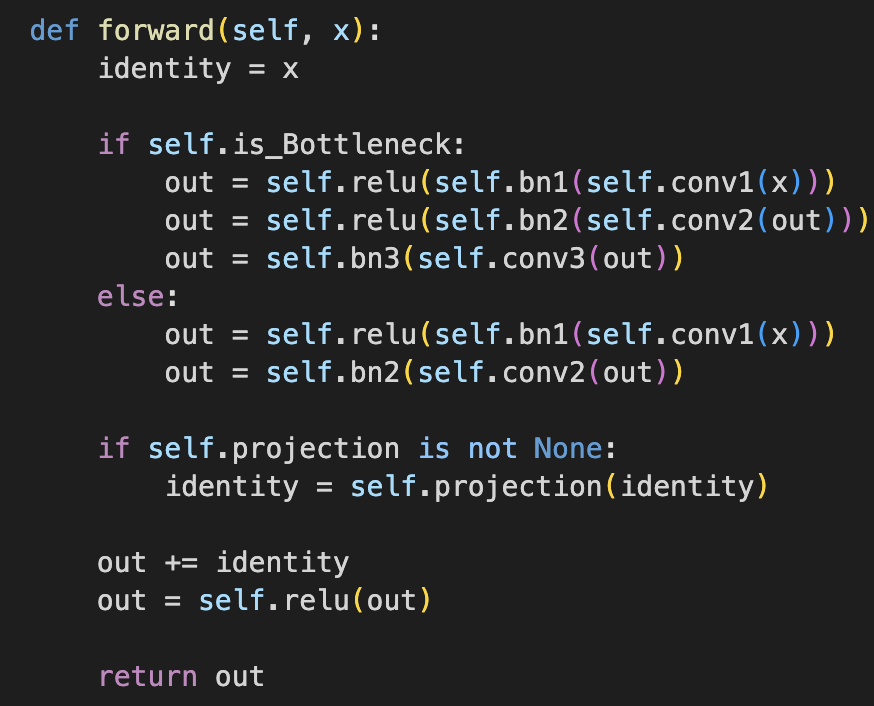
The above is the bottleneck initialization where expansion represents the bottleneck expansion factor mentioned in the model parameter.

is\_Bottleneck defines if the block is a basic block or a bottleneck block and is a boolean value. If the version is lower than ResNet50 this should be false because they are using basic block. But all other version with more layers including ResNet50 itself should use bottleneck block.

Finally, the identity will check if the input and output channel matches, if not there will be modification needed. The modification will be made through 1\*1 convolution followed by batch normalization, and the purpose is to resize the input so it can be added to the block’s output.

We have used ReLu activation function in our model to introduce the non-lineariry. If we are activating the bottleneck block, the block should be defined with 3 convolutional layers (1\*1, 3\*3, 1\*1). Otherwise, a 2 convolutional layer block will be applied(2 of 3\*3 convolutional layers). All convolutional layers are followed by the batch normalization to stablize the learning process.

**Forward pass:**

****

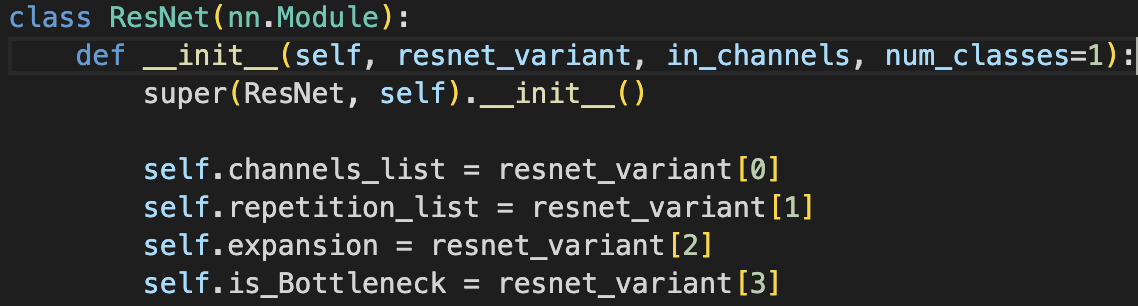
The identity is set up in the beginning and will be added back to the output of the residual block to create the skip connection which allows the gradient to flow directly through the network. The purpose is to encounter the vanishing gradient problem.

Similar logic is applied to check if bottleneck block is being used, the input is passed through the first convolutional layers, and then batch normalized and finally passed through the ReLu activation function. After the final convolutional layer in the block, an output should be generated.

Then we need to check the dimension of the output and identity are equal, because we need to add identity back to the output. If the projection is defined, it means the dimensions are not the same, and we need to resize the identity to match with the output.

Finally, we identity back to the output and pass the output through ReLu activation function to get our output from the block.

**ResNet Model:**



Different from other ResNet models, usually they are designed for multi-classification tasks, as we wanted to implement them in binary classification, the number of output neuron(num\_classes) needs to be adjust to 1 and followed by the sigmoid activation function implemented in the output of the final fully connected layer

The initialization is then defined using the model parameter and stored in a list called resnet\_variant. Then the model start with it’s initial convolution and pooling so as other CNN models. Followed by the ResNet block that is defined by if the block is basic block or bottle neck block(4 blocks are applied). After that, an average pooling and output from final fully connected layer is applied.

Finally, in the forward pass method, we applied all the operation that is defined in the initialization model.

**Training:**

# 

Since our model focus on binary classification task, the loss function we use is Binary Cross Entropy, and we choose Adam as our optimizer to apply the addaptive learning rate to avoid gradient going too fast to lose the local minima or too slow to be inefficient(high computational cost).

# **6.0 Baseline Model**

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 (Bhagwat, 2021). The structure of the AlexNet can be referred to Figure 3

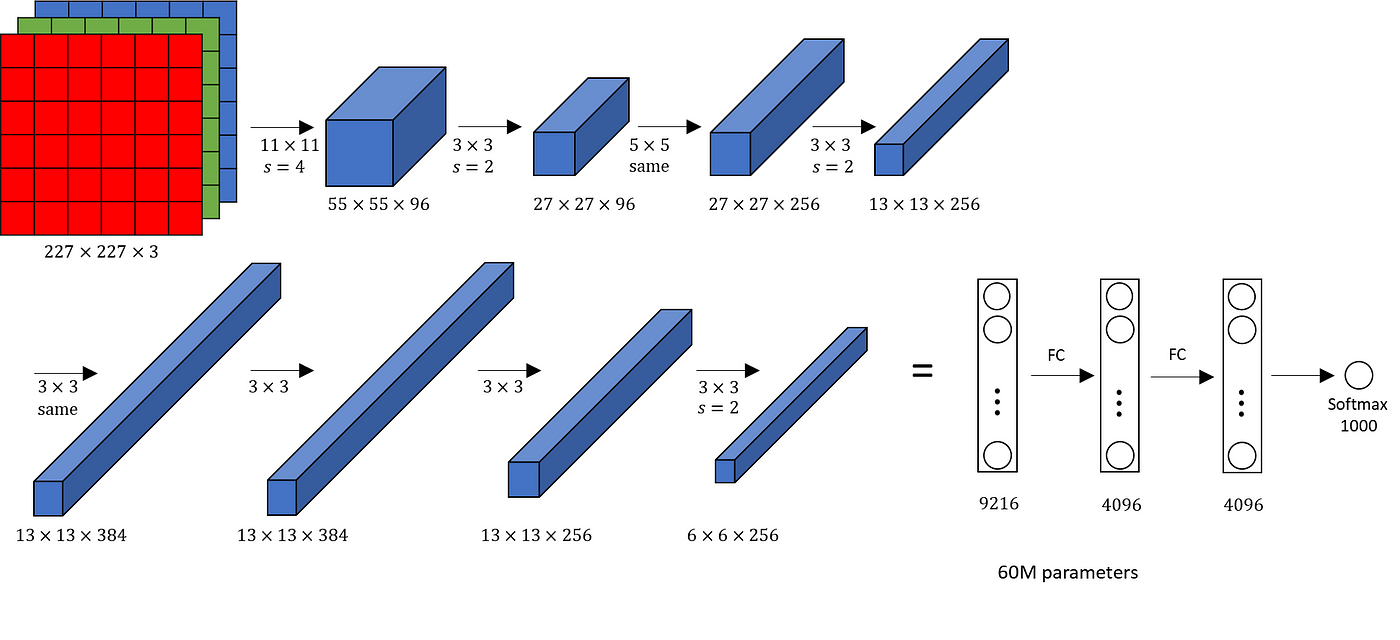


Figure 3: AlexNet Structure

**6.1 THE CHOICE OF ALEXNET:**

AlexNet is a large, deep convolutional neural network to classify 1.2 million high-resolution images and won the ImageNet LSVRC-2010 contest into the 1000 different classes. Convolutional neural networks (CNNs) can make strong and mostly correct assumptions about the nature of images (namely, stationary statistics and locality of pixel dependencies). As a beginning model of DeepLearning, it’s a great starting 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 (Krizhevsky et al., 2012), the primary model of this project will build a CNN model that imitates the GoogleLeNet with residual learning and building a deeper layer model for this binary classification task.

**6.2 Configuration/Structure of Baseline**

The initial preprocessing step involved rescaling the pixel values of the input images to a range between 0 and 1. The AlexNet model expects input images with dimensions of 227 x 227 pixels and 3 RGB color channels.

The model architecture consists of 5 convolutional layers followed by 3 fully connected layers. ReLU activation function was applied to all convolutional and fully connected layers. Additionally, max pooling layers were inserted after the first, second, and fifth convolutional layers to downsample the feature maps. Batch normalization was applied after the first two convolutional layers to improve training stability.

To mitigate overfitting, dropout was applied to the three fully connected layers. The output layer employs a softmax activation function with 2 units corresponding to the number of classes.

During training, the model utilized the Adam optimizer with a learning rate of 0.001 and categorical cross-entropy loss function, which can be adjusted for multi-class classification tasks. The baseline model was trained using a batch size of 256 for a total of 10 epochs.

# **7.0 Result**

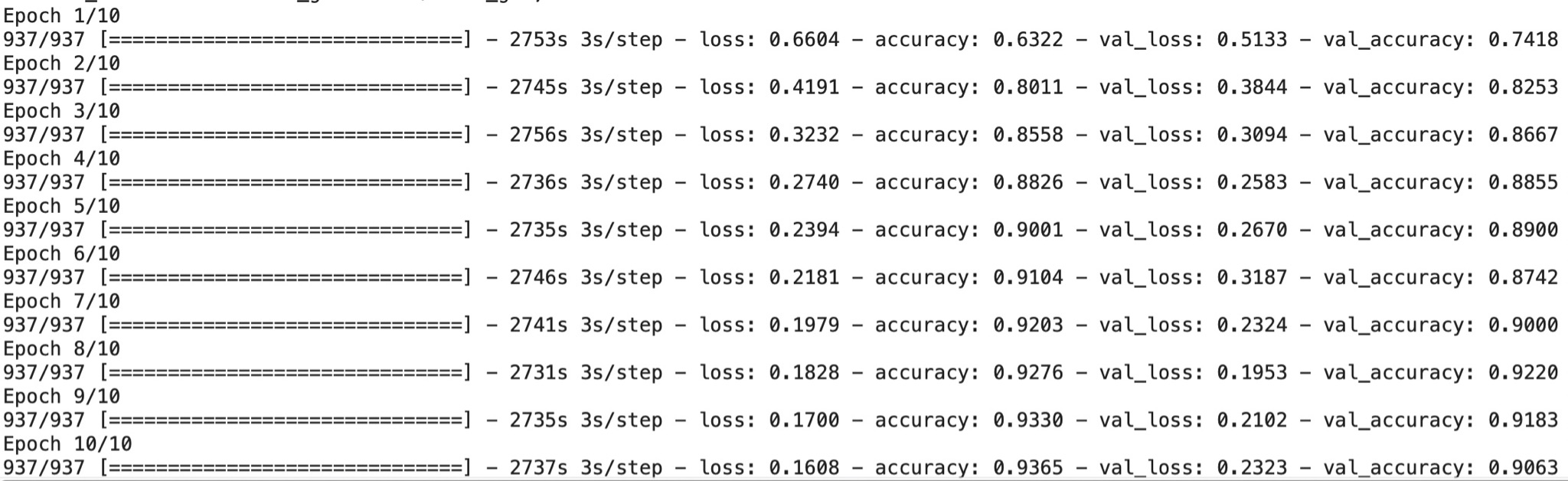
## **7.1 Quantitative Results:**

In order to compare between 2 models’ performance. The main criterias applied were accuracy and loss of 2 models. Based on the accuracy and loss from model training, few assumptions were made to calculate the F1 score.

**Baseline model:**

The baseline model achieved its highest training accuracy of 93.65% at the 10th epoch, while the validation accuracy peaked at 92.20% during the 8th epoch. This difference suggests the potential overfitting of the model. To address this overfitting, the model will select the weights from epoch 8. At this point, the model required 2731 seconds for computation, with each step taking approximately 3 seconds. The training loss was recorded at 0.1828, with a corresponding training accuracy of 92.76%. Validation loss was at 0.1953, accompanied by a validation accuracy of 92.20%. Based on calculations using data from epoch 8, the model achieved an F1 score of 0.925 with precision of 0.9276 and recall of 0.9224,indicating a good balance between precision and recall . The model is effective at both identifying real images (precision) and capturing most of the deepfake images in the dataset (recall).

The similar accuracy of training and validation dataset suggests that the model generalizes well to unseen data.The F1 score shows that the baseline model is accurate and reliable in identifying the positive class while avoiding misclassifications, creating a solid base for the project model to compare with.

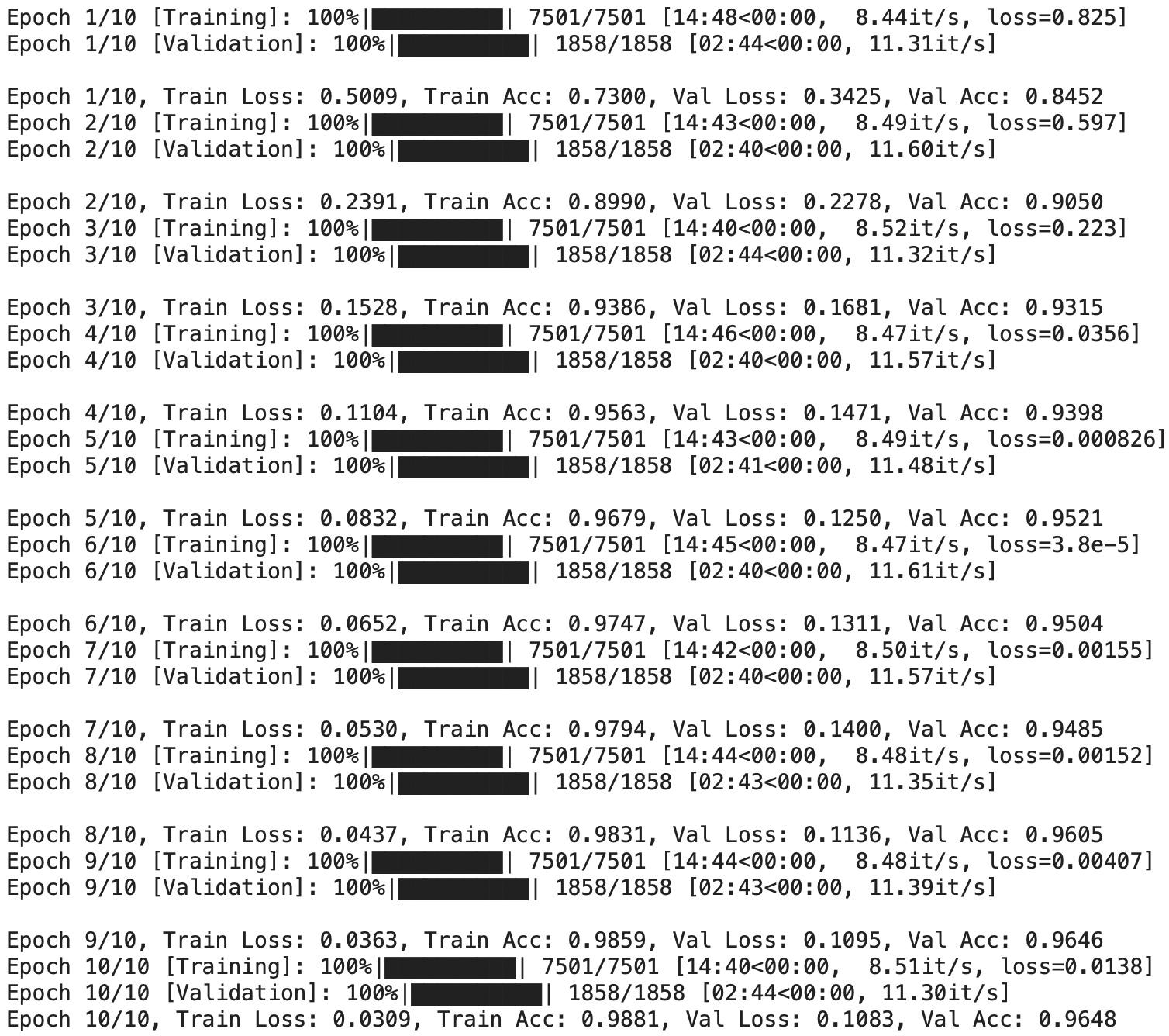


**ResNet50:**

The ResNet model reached its peak performance at epoch 10, achieving a training loss of 0.0309 with a corresponding training accuracy of 98.81%. The validation loss was measured at 0.1083, accompanied by a validation accuracy of 96.48%. Compared to the baseline model, the ResNet model demonstrated significant improvements. The training accuracy increased by approximately 6%, while the validation accuracy improved by around 4%. Moreover, there was further improvement in both training and validation loss, with the training loss decreasing from 0.1828 (baseline) to 0.0309 and the validation loss dropping from 0.1953 (baseline) to 0.1083. Overall, there was a remarkable 50% reduction in dataset loss.

Upon evaluation, the ResNet model shows outstanding precision and recall scores, both calculated at 0.9645. This balanced performance is further reflected in the F1 score, which also stands at 0.9645. Notably, all three scores show an improvement of 0.04 compared to the corresponding scores of the baseline model.

These enhancements demonstrate the ResNet model's superior ability to accurately classify images, leading to improved performance and efficiency compared to the baseline model. The substantial reduction in loss and the balanced precision, recall, and F1 score demonstrate the model's effectiveness in handling the dataset and its capability to generalize well to unseen data.



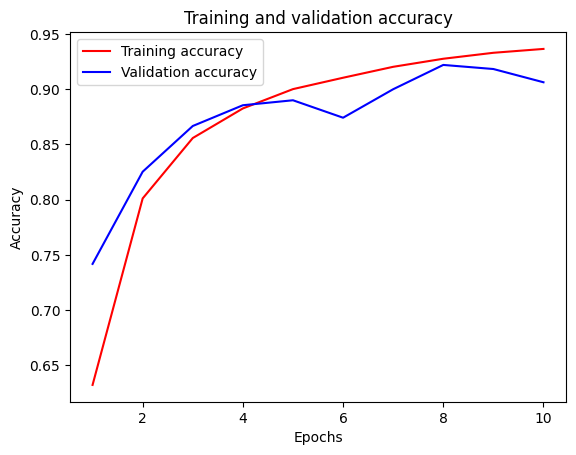
## **7.2 Qualitative Results**

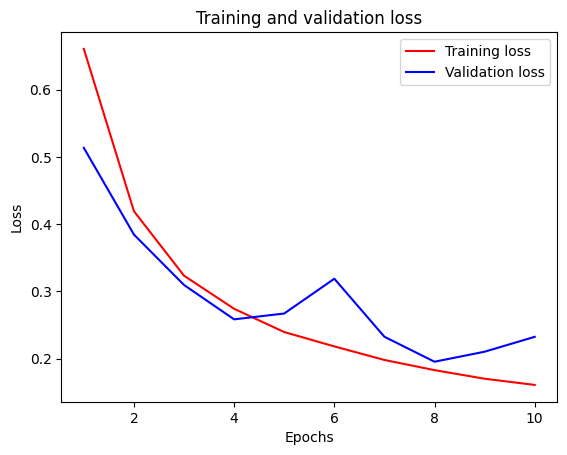
**Baseline model:**

The loss and accuracy graphs of the 10 epochs from the baseline model reveal valuable insights. As the number of epochs increases, a noticeable gap emerges between the training and validation accuracy lines, indicating the potential onset of overfitting. To address this issue, several adjustments can be made. Decreasing the learning rate or increasing the batch size are potential strategies, while extending the number of epochs could also help monitor any subsequent changes in accuracy. Additional strategies like regularization, data augmentation have been applied.

Given the existing dataset, implementing early stopping could be beneficial. By tracking the validation loss, particularly as of epoch 8, we can identify when the model's performance begins to degrade and stop training to prevent further overfitting.

With a finalized accuracy of 92.2%, further analysis on errors can provide valuable insights. Extracting sample errors from the test dataset and exploring common features that haven’t been effectively captured by the model can be insightful. Features such as teeth and fingers, which are typically challenging for machine learning models to generate or identify, may contribute to errors. To enhance accuracy, adding more layers to enable deeper feature exploration by the model could be beneficial.



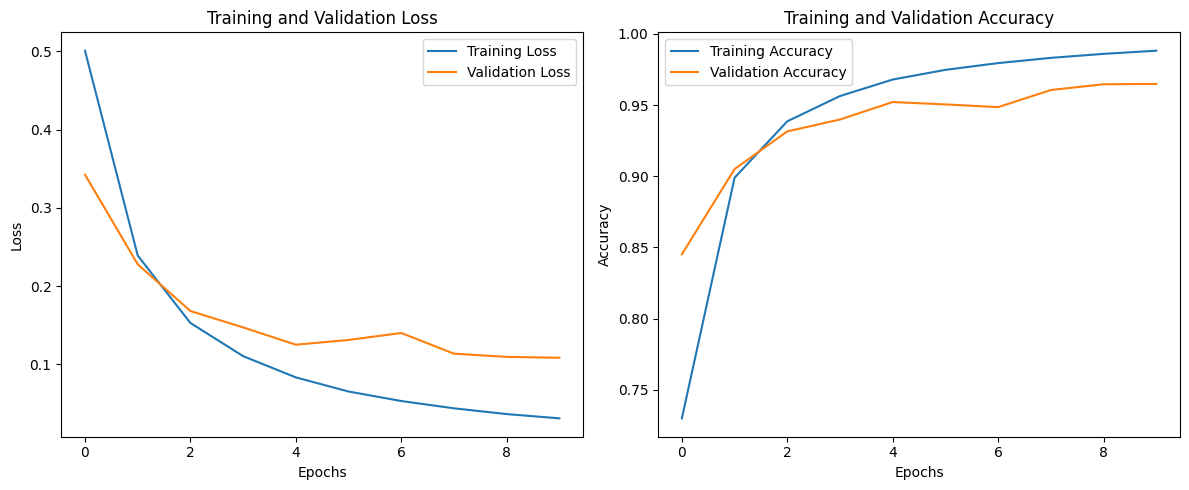


**ResNet50:**

The graph of ResNet shows a continuous improvement in accuracy and a decrease in loss as the training epochs progress. In contrast, the baseline model exhibits an increase in loss over time. To further enhance accuracy and optimize the model's weights, additional epochs can be executed.

With the growing gap between the training and validation loss, there is a growing risk of overfit with more epochs. Similar strategies employed for the baseline model can be applied, such as reducing the learning rate and increasing the batch size. Given that the learning rate has already reached 0.001 and the batch size is currently set at 32, adjusting the batch size could be the initial parameter to consider. Larger batch sizes facilitate smoother gradient updates, aiding in better generalization by averaging out noisy gradients.

Despite the smaller batch size compared to the baseline model (which had a batch size of 256), ResNet 50 boasts a significant advantage in computation time. This efficiency is a direct result of the smaller batch size and underscores the benefits of this approach. It’s crucial to find the balance between model complexity, accuracy and computational cost.



# **8.0 Evaluate model on new data**

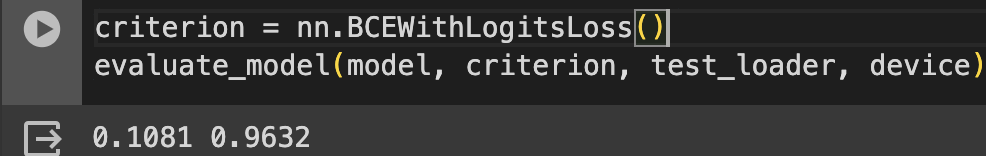
Describe the efforts taken to ensure the results are a good representation of the model’s performance on new data. Can you evaluate model on new data? This will depend greatly on the problem being solved.

◦ 10/10 Team is able to obtain new samples that have not been examined or used in any way to influence the tuning of hyperparameters. Model performance meets or exceeds expectations for the problem being solved.

In order to achieve the performance of model on new data, there are two aspect that we need to consider. The first thing first is the data processing, our team has taken effort into selecting data. The data must be clean. It means that in the evaluation phase we are testing on the test dataset, there could not be images that has appeared in the training and validation dataset. To avoid this possible instance, before our team has split the data into train/validation/test dataset, our team went over the image to make sure we eliminate all the duplicates. Secondly, the team need to make sure all the labels on the images are correct to avoid misinformation feeding to the model.

The second aspect is the model itself. Even though the task the model focus on is the binary classification task, the deepfake image itself is complex(pixel wise) because there’re facial emotions, postures and features and patterns are used to determine if the image is generated by AI. Therefore, our team has searched for the existing models that can handle classification tasks with complexity. However, while looking into the performance of the model outcome, we also need to consider our capabilities. We cannot make a model that cost high in time and computational resources. Our team has tried many intermediary models like GoogleNet, ResNet17 and ResNet50. According to the result, both GoogleNet and ResNet 50 did not match the expectation in the validation accuracy, so they are abandoned. ResNet50 has achieved a validation accuracy of more than 96% after some tuning on the hyperparameters. However, due to the high computational cost, we do not have the budget to tune the hyperparameters further.

Evaluation result:



Through an evaluation method, our team has pass the clean and unseen image data from the test loader and has achieved a loss of 0.1081 and accuracy of 96.32% which is similar but slightly lower than the validation accuracy and the loss is also lightly lower. A 96.32% test accuracy of classifying deepfake is considered a success taking into the limited time and resources that our team has. However, there are still huge progress that could be made. If we want to improve the performance even more, we can try tuning various hyperparameters in the bottleneck block, or we can try models with higher complexities but require more computational costs(ResNet101 and ResNet152).

# **9.0 Discussion**

**Architectural Efficiency and Depth**

The superior performance of the ResNet model, as evidenced by our experimental outcomes, underscores the critical role of architecture depth and efficiency in handling high-dimensional data. The inherent design of ResNet, characterized by its residual blocks and bottleneck features, adeptly mitigates the vanishing gradient problem, thereby facilitating the effective learning of deep representations. This architectural nuance is particularly crucial in applications like deepfake detection, where the model must identify subtle visual cues indicative of manipulation. The question then arises: How can we continue to enhance architectural efficiency to cater to increasingly sophisticated manipulations without exponentially increasing computational demand?

**Balancing Computational Cost and Model Complexity**

Our project's pragmatic approach to selecting ResNet50, over potentially more complex but computationally demanding alternatives like ResNet101 or ResNet152, highlights an essential consideration in machine learning: the balance between model complexity and computational efficiency. This balance is not merely a technical consideration but a strategic decision that impacts the scalability and accessibility of machine learning solutions across different sectors. As professionals, we must navigate these considerations, optimizing for both performance and practicality. Future research could explore adaptive architectures that dynamically adjust complexity based on the task's requirements or the available computational resources.

**Rigorous Data Management and Model Evaluation**

Another critical learning from our project is the paramount importance of rigorous data management and methodical model evaluation strategies. By ensuring a clean, non-overlapping dataset for training, validation, and testing, we have laid a foundational basis for assessing the true performance of our model. This meticulous approach is indispensable, especially in fields where model reliability and generalizability are crucial. It prompts a broader discussion on the standards for data integrity and the methodologies for transparent, reproducible model evaluation in the machine learning community.

Moreover, the continuous evolution of deepfake technologies necessitates ongoing research into more advanced detection methods. This could include exploring multimodal detection systems that combine visual cues with other data types, such as audio, to improve detection accuracy. Additionally, leveraging unsupervised learning or semi-supervised learning approaches to cope with the rapidly expanding variety of deepfake techniques represents a promising research direction.

# **10.0 Ethical Considerations**

To ensure the integrity and ethical application of our AI advancements, particularly in the realm of deepfake detection, our project is anchored by a robust ethical framework. This framework is designed to respect and protect individual autonomy, privacy, and the equitable treatment of all demographics. It is outlined as follows:

**1. Explicit Consent for Data Use**

We prioritize obtaining explicit consent from individuals before utilizing their personal images, videos, or recordings for AI training. Our approach is rooted in the respect for personal autonomy, ensuring that data is used only with the clear and informed agreement of those involved.

**2. Rigorous Privacy Protections**

The privacy of individuals whose data we handle is of utmost importance. We are committed to implementing measures such as data anonymization, restricting data access to authorized personnel only, and adhering to strict privacy regulations like the GDPR. Our dedication extends beyond legal compliance, embracing a moral obligation to protect personal information with the highest standard of care.

**3. Equity and Bias Reduction**

We acknowledge the risks of bias within datasets that fail to represent all demographics fairly. To combat this, our project is committed to adopting a diverse data approach. By ensuring our AI models are trained on a wide range of data, we aim to achieve equitable performance across various groups, thereby tackling the issue of bias in AI technologies.

**4. Transparency and Accountability**

Our project upholds a commitment to transparency regarding how data is collected, used, and safeguarded. Recognizing the importance of accountability, we are dedicated to admitting and rectifying any errors in our processes. This transparency is vital for maintaining public trust and demonstrates our commitment to ethical standards that respect individual dignity and rights.

Through this ethical framework, our project aspires not only to enhance the technical capabilities of AI in detecting deepfakes but also to contribute positively to the digital environment by upholding the highest ethical standards. This approach ensures that our advancements in AI technology are both technically sound and morally responsible.

# **11.0 Project Difficulty / Quality**

A measure of how “difficult” the project is, and how well your model performs given the difficulty of your problem. If your problem is more difficult than what one might expect, you should clearly articulate why in the body of your report.

◦ 6/6 Team creates a model that performs better than expected on a challenging project. Team demonstrates learning beyond the requirements of (e.g.,) the labs.

Our team think the project has a mid-high difficulty level. Since we are allowed to decide what topic we want to focus on, the scope is for us to define. As we have chosen a binary classification task as our focus, it lowers the difficulty level since there is a higher chance for the model to “guess” the result correctly. However, deepfake image itself is complex. All the colour, organs, facial expressions, details and postures makes the “fake” or “real” images. The complexity of the image means higher dimensionalities in the images result in an increase in the computational complexity and the amount of data needed to train the model without overfitting. The variation of the images is also a concern, variation in color, the way images are made, and background light can vary the image significantly. Since the image data we have chosen contain different backgrounds, these irrelevant information requires the model to distinguish between the relevant and irrelevant informations in highly complexed images.

Given the complexity of images, it is difficult to achieve a high accuracy since it requires the strong ability for model to learn and identify from the subtle differences from the complex images. To encounter the misinformation deepfake technology has broughten to the media, our goal is to achieve an accuracy rate of 90% or more. Therefore, we used AlexNet as our baseline model. Achieving high accuracy while given complex input for classification is difficult. Therefore, instead of using basic CNNs, we need to find more powerful neural network architectures that can handle complex cases and out perform our baseline model. ResNet50 was finally decided to be our primary model, however, the structure and model parameters need adjustment to fit in our tasks. Our team has spend many time in debugging the model because we need to find the source of the bug and fix it. The most difficult part is the consistency of dimensions and maintaining compatibility through all layers of the model. As the input image passes through all the layers and blocks and comes to an output, it takes effort and experience to identify where the source of the bug comes from and what is the cost of it.

In conclusion, the binary classification makes our project mid level difficulty. However, the complexity of images and models and the requirement of large amount of data with the goal of needing to achieve high accuracy make our project more difficult than expected. The good side of it is with the whole team effort, we are able to achieve a test accuracy that we originally expected, and has successfully implemented complex models that are more difficult than the course requirement.

**12.0 Link to GitHub/Colab Notebook**

# **13.0 Reference**

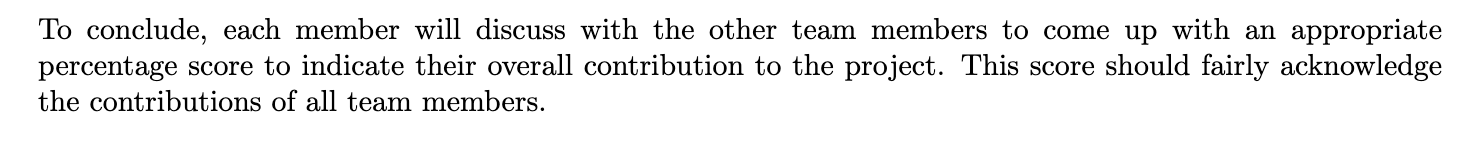
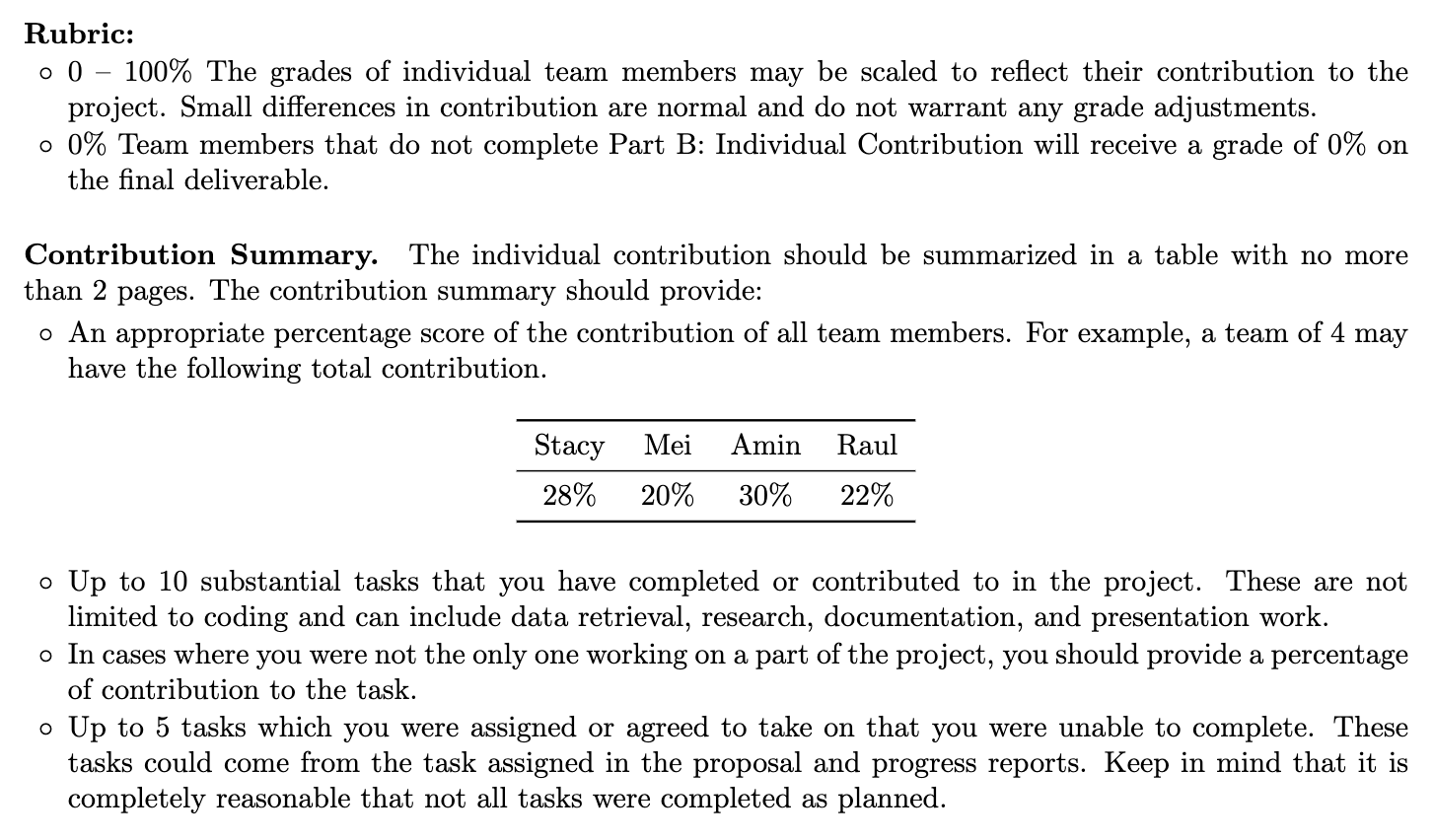
非特定章节要求：

**• Structure, Grammar & Mechanics (8 points)**: We are looking for a document that is easy to follow, grammatically correct, and well-written. The document must be written using Latex based on the course template.

◦ 8/8 Clear, concise, and well-written document. Exceeds expectations.

**给个模板，这个部分是各自写一份提交**

**Part B Individual Contribution to Project (2-page limit)**



| Team Member | Vincent | Shawn | Isabel | Joey |
| --- | --- | --- | --- | --- |
| Contribution | 25% | 25% | 25% | 25% |

**Vincent:**

**Tasks Completed:**

* **Deepfake Detection Model Research:** Identified and researched suitable modelling techniques for Deepfake detection.
* **Image Dataset Collection:** Collected a substantial dataset of over 300,000 images for analysis.
* **CNN Model Architecture Construction:** Led the design and development of the CNN model architecture.
* **Model Parameters Optimization:** Conducted thorough tuning of the model parameters for optimal performance.
* **ResNet Framework Enhancement:** Focused on enhancing the ResNet framework to refine the final model's accuracy.
* **Performance Analysis:** Analyzed and provided insights from the Baseline model's results.

**Tasks Cancelled:**

**Isabel:**

**Tasks Completed:**

* **Image Dataset Collection:** Collected a critical initial dataset for the project.
* **Background Research:** Researched deepfake detection methods and CNN technologies.
* **Baseline Model Architecture Construction:** Developed the baseline model for the project.
* **Model Parameters Optimization:** Tuned the model parameters for enhanced performance.
* **Performance Analysis:** Analyzed and provided insights from the Baseline model's results.
* **Documentation:** Assisted in composing the final report with detailed documentation.

**Tasks Cancelled:**

**Shawn:**

**Tasks Completed:**

* **Project Management:** Directed the project management to ensure adherence to deadlines.
* **Image Dataset Collection:** Supported the data collection process for project analysis.
* **Data Processing:** Assisted in data cleaning for dataset integrity.
* **Model Development Collaboration:** Co-constructed the baseline model, setting the stage for further development.
* **Final Integration:** Handled the integration and submission of the final model to the required platform.

**Tasks Cancelled:**

* **Model Tuning (Re-assigned):** Tasked with collaborating on the model's fine-tuning process alongside Vincent to enhance performance. Due to asynchronous scheduling conflicts and technical constraints, this responsibility was fully transferred to Vincent.
* **CNN Model Development (Adjusted Participation):** Initially involved in the early stages of developing the CNN model, aiming to contribute to both the construction and testing phases. Technical barriers limited accessibility for testing, leading to transition to assisting with other members' tasks, focusing on areas where his contribution could be more effectively applied.

**Joey:**

**Tasks Completed:**

* **Background Research:** Reviewed relevant literature to support the project's foundation.
* **Data Processing:**Maintained data integrity through rigorous cleaning.
* **Model Development Collaboration:** Co-constructed the baseline model, setting the stage for further development.
* **Model Performance Monitoring and Reporting**: Monitored the model's performance, reporting on essential metrics.
* **Documentation:** Assisted in the detailed composition of the project's final report.

**Tasks Cancelled:**