Concordia University
Engineering and Computer Science

COMP 6721 – Applied Artificial Intelligence - Fall 2023

Project Assignment Part 1 & 2

Team Members
Ronak Patel, Student ID: 40221814 (Training Specialist)
Vijendra Ahirwal, Student ID: 40221273 (Evaluation Specialist)
Aayush Patel, Student ID: 40272388 (Data Specialist)

Project Repository:

`https://github.com/vijendraahirwal/AppliedAI`

Contents

1 Dataset

1.1 Dataset Information

In this report section, we are providing an overall view of the datasets utilised in the project, and its characteristics such as highlighting the total number of images, and special features of each dataset. We employed two different ways to obtain the final data for the project.

- **Downloaded from Kaggle** The dataset used in our project comprises grayscale images of faces, each with a resolution of 48x48 pixels. This dataset has been prepared by Pierre-Luc Carrier and Aaron Courville as part of an ongoing research project.[2] Their generosity in providing a preliminary version of this dataset to the public has been contributing to advancements in the field of emotion recognition from facial expressions.

  - **Face Expression Recognition Dataset** [2]

- **Bulk downloaded from Search Engines (Google, Bing)** In the course of our project, it became evident that two distinct images classes, "focused" and "bored," were essential for our project to be successful but were not represented in the FER 2013 dataset, our initial source of facial expression data. To address this issue, we undertook a data scraping process by bulk downloading facial images using specialized software from popular search engines such as Google Images and Bing Images [8]. The acquired images were then properly processed to ensure they met the criteria necessary for integration into our project.

Types of Image Classes

- **Neutral Expression Class**: This image class comprises images of individuals displaying neutral or emotionless facial expressions. These expressions typically lack any obvious emotional cues. There are total 552 images present in this class. This class images were directly sourced from FER2013 [2] dataset. The dataset aims to represent a broad spectrum of ethnicity to ensure a comprehensive and unbiased representation of neutral facial expressions.

- **Bored Expression Class**: This image class encompasses images of individuals who are manifesting boredom through their facial expressions. Boredom is characterized by a lack of interest or engagement, often denoted by drowsy or disinterested facial features. There are total 537 images present in this class. The images within this class have been sourced from a variety of sources, including publicly available image databases, and online image search engines under creative common licence.

- **Focused Expression Class**: This image class contains images of individuals who are deeply engrossed in a task, study, reflecting intense concentration and engagement through their facial expressions. There are total 572 images sourced

for this class. The images within this class have been sourced from a variety of sources, including publicly available image databases, and online image search engines under creative common licence.

- **Angry Expression Class**: This image class includes images of individuals who are visibly displaying anger through their facial expressions. Anger is characterized by features such as furrowed brows, clenched jaws, and intensive staring. There are total 638 images sourced for this class. This class images were directly sourced from FER2013 dataset.

1.2 Dataset Justification and Challenges Faced

In our project, we carefully selected specific datasets to address the unique requirements of our project. The selection of these datasets was based on their relevance to the project and their accessibility under suitable licensing terms. We encountered distinct challenges during this process, particularly in obtaining images for the "Bored" and "Focused" expression categories.

Why FER2013?

The selection of the FER2013 dataset for the "Angry" and "Neutral" expression classes was guided by the following reasons:

- **Public License and Recognition**:The FER 2013 dataset is available under a public license, making it accessible for research purposes without legal constraints. It is a well-known and widely recognized dataset within the AI community, known for its contribution in advancing the field of emotion recognition
- **Community Acceptance**:The dataset's track record, with over 50k downloads, signifies broad acceptance and validation within the AI community. This extensive usage history validated its suitability for our project.

Obtaining FER2013 was relatively straightforward. The dataset was readily available online, and we could easily download it.

Why Bulk Downloading?

After a very extensive research we came to the conclusion that we will employ the data scraping technique to find credible images for these two categories. To obtain images for the 'Bored' and 'Focused' categories, we scraped images from Google [3] and Bing [8]. We used a free software tool [16] for the images scraping purpose. These images have Creative Commons licenses. Therefore, there was no any issue with the copyright of the images. In total, we scraped more than 1500 images across both categories. To get them in shape, we employed extensive data cleaning and data processing techniques, which will be explained later in report. We faced a lot of challenges while gathering these two

classes images.

Some of them are as follows:

- **Data Acquisition**: The unavailability of pre-existing datasets for "Bored" and "Focused" expressions led us to perform bulk downloads from search engines like Google and Bing. The challenge was to locate images that accurately depicted the desired emotional states.
- **Licensing and Copyright**:Ensuring that the downloaded images were available under Creative Commons licenses was crucial to avoid copyright issues.
- **Data Quality and Diversity**:After downloading, we had to carefully filter images to meet the minimum required number for each category while ensuring they adhered to good dataset guidelines. We encountered variations in dimensions, lighting conditions, and the presence of diverse ethnic backgrounds.
- **Non-Human Faces**:We also had to identify and remove images that did not contain human faces, maintaining the dataset's quality and relevance.

These challenges underline the careful efforts involved in obtaining and preparing the "Bored" and "Focused" expression images. Despite the obstacles faced, the outcome was a dataset that expanded the emotional range of our project, facilitating a more comprehensive analysis of human emotion recognition from facial expressions.

2 Data Cleaning

This section presents an overview of the necessity for data cleaning, the various techniques utilized for data cleaning in the project, and the challenges encountered during the data cleaning process.

Why Data Cleaning in the First Place?

Data cleaning is essential in our project for designing CNN models to classify facial expression images into four different categories (Focused, Bored, Angry, Neutral). We sourced our dataset in two distinct ways, from Kaggle through downloads and from Google and Bing image searches through scraping.

The need for data cleaning can be explained as follows:

- **Data Consistency**: When we download datasets from various sources, they may have inconsistencies in terms of image dimensions, image quality, and file formats. Data cleaning helps ensure that all images are standardized to a common resolution, format, and quality, enabling the CNN models to process the data consistently.
- **Outliers and Noise**: The downloaded dataset may contain outliers or noisy images that do not belong to the desired emotion categories. Data cleaning involves the identification and removal of such outliers to maintain dataset purity.
- **Labeling**: Data cleaning verifies the accuracy of these annotations and rectifies any labeling errors.

2.1 Data Cleaning Process and Challenges

Step 1: Filtering Unnecessary Images

- **Challenge**: From downloaded images, determining which images contain human faces programtically.
- **Solution**: We utilized a Python script to address this challenge, automatically removing images that did not contain human faces, ensuring the dataset's relevance.

Step 2: Locating Faces in Images

- **Challenge**: Selectively isolating only the frontal face of subjects, as our CNN's objective is to detect facial expressions.
- **Solution**: To address this challenge, we implemented a Python script that cropped and extracted only the facial regions from the entire images, reducing processing load and improving model performance. We utilised HAAR classifier to perform face detection.[7] Figure 1 is showing the effects of the step 1 and 2 [13].

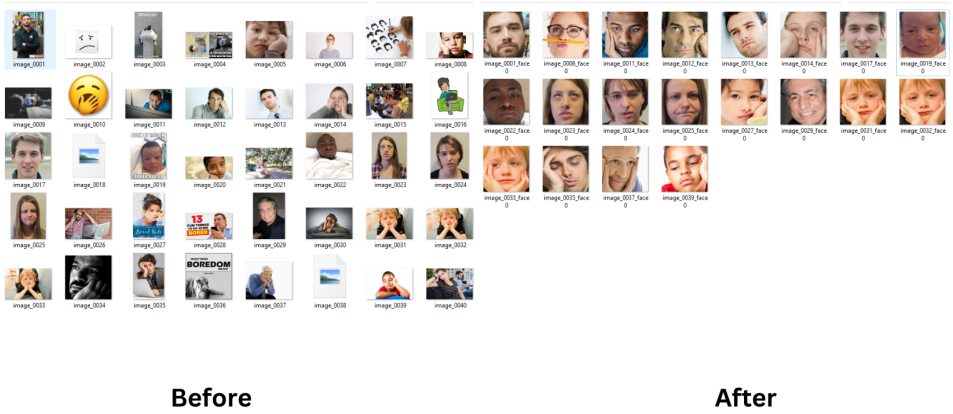


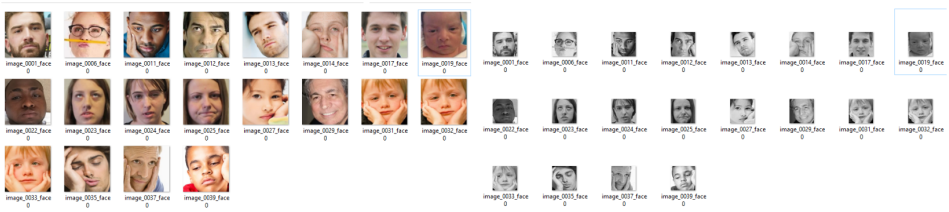
Figure 1: Effect of Step 1 and 2

Step 3: Visual Inspection of Images

- **Challenge**: Images from two categories, "Angry" and "Neutral," were originally 48x48 pixel grayscale images, while the other two categories, "Bored" and "Focused," had arbitrary dimensions and color channels due to downloading from the internet.
- **Solution**: To standardize the dataset, a Python script was employed to resize and convert all images to a consistent 48x48 pixel grayscale format, ensuring uniformity in data dimensions and color channels. Figure 2 is showing the effects of the step 3.

Step 4: Image Sharpening

- **Challenge**: To enhance the quality of the images for better learning during CNN training, we aimed to make the edges in the images more prominent. Edges in images play a crucial role in identifying and characterizing facial expressions. However, the challenge was to achieve this enhancement without amplifying noise content within the images.
- **Solution**: To address this challenge, we implemented a two-step process. Firstly, we applied image blurring to the dataset. Blurring was employed as a preliminary step to reduce noise and any unwanted artifacts in the images. Subsequently,



Before **After**

Figure 2: Effect of Step 3

Before **After**

Figure 3: Effect of Step 4 All Photos

Figure 4: Effect of Step 4 Individual Photo (Zoomed Version)

we employed a technique known as "unsharp masking" to transform the blurred images into sharpened ones. Figure 3 and 4 is showing the effects of the step 4.

- **A Little About Unsharp Masking**
  - Unsharp masking is an image processing technique that involves subtracting a blurred version of the image from the original image to enhance edges and fine details. It works by creating a "mask" that highlights areas of contrast and sharp transitions in the image. This process effectively increases the emphasis on edges, making them more prominent and contributing to better model training. Unsharp masking is a valuable tool for improving image quality and can be particularly useful in tasks like facial expression recognition, where capturing subtle details is vital for accurate classification. For visual inspection of the effect of this technique, please refer to Figure 5 [12]

Upon completing the aforementioned procedures, we arrived at a stage where our images were appropriately prepared for utilization by CNN.

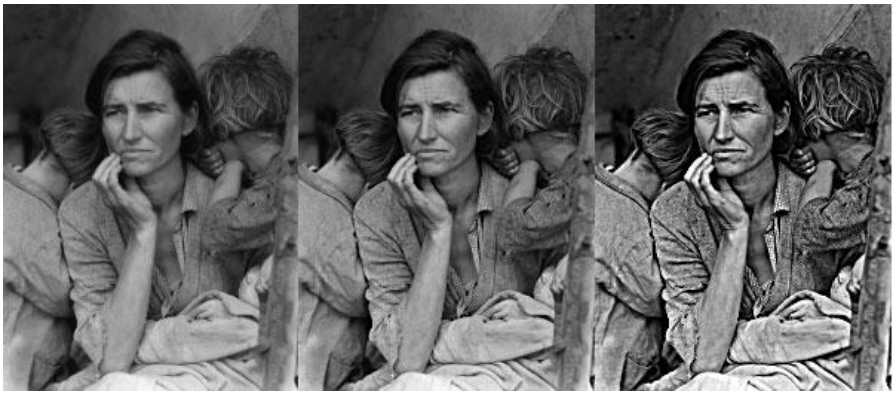


Figure 5: Sample Image Showing The Effect of Unsharp Masking

3 Labeling

In the process of organizing the gathered images for our project, we initiated by establishing four distinct directories, each corresponding to one of the four emotion categories: Bored, Focused, Angry, and Neutral. Subsequently, all images were systematically placed into their respective directories based on their associated emotional expressions.

Given that images from different emotion categories were sourced from diverse origins, we recognized the need for a standardized naming convention that would prove valuable for future model training and validation requirements. To address this, we implemented a Python script that systematically traversed each directory and applied a renaming procedure to individual images. The new nomenclature adopted for each image adhered to the format of categoryName_XXXX.jpg, where "categoryName" represented the specific facial emotion category, and "XXXX" signified the image's unique sequence number within that category.

This approach resulted in the establishment of a universal naming convention across all images, thereby facilitating the subsequent phases of model training and evaluation. For these phases, our plan is to use the first word of each image's name as its associated label, easing the integration of these images into the training and evaluation processes of our project. Please refer to Figure 6

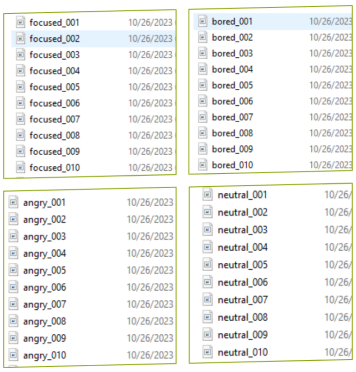


Figure 6: Images Labels After Proper Labeling

4 Dataset Visualization

Using Matplotlib, we carried out the following visualization tasks.[15]

4.1 Class Distribution

In this section, we employ a bar graph to showcase the distribution of images across each class within the dataset. This visualization serves as a important assessment of the dataset's class balance, offering a quick overview of whether each class contains a reasonably equal number of samples. Detecting significant imbalances among classes is vital, as it can significantly impact the model's performance. Achieving a balanced dataset is crucial to ensure the machine learning model is trained effectively. Figure 7 [9]

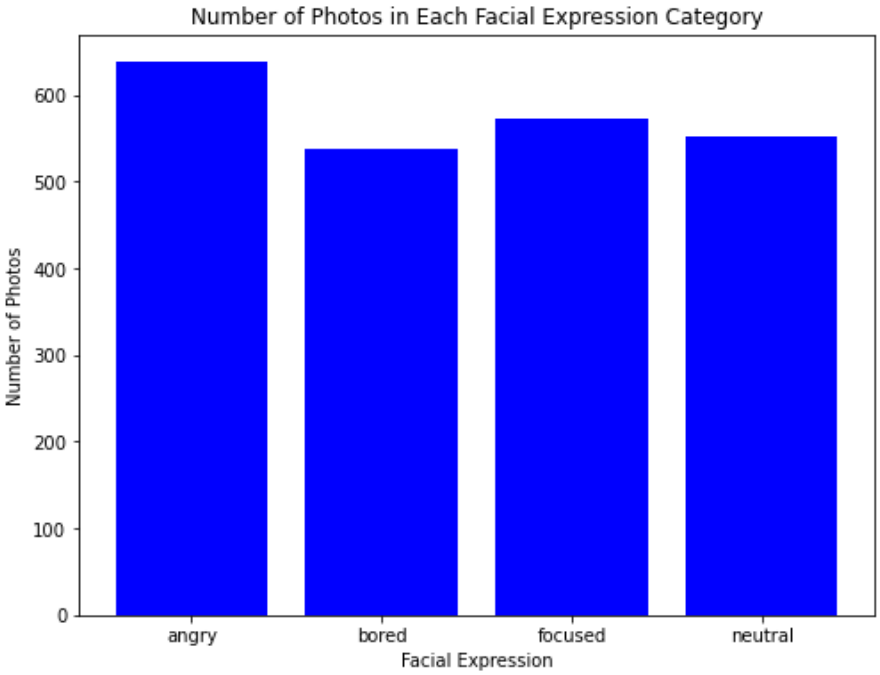


Figure 7: Class Distribution of Each Category

4.2 Sample Images

To provide a more concrete understanding of the dataset's content, we present a curated selection of 25 random images in a 5x5 grid. This visual representation allows us to see the dataset's diversity, enabling the identification of anomalies or potential errors. Please refer to Figure 8

4.3 Pixel Intensity Distribution

In this section, we delve into the pixel intensity distribution within the dataset. By generating histograms of pixel intensities for a randomly selected set of images, we gain insights into the lighting conditions and the gray level distributions. Figure 10
We would like mention that to learn LaTex to make this report structure we utilised ChatGPT. [11]

Figure 8: Random 25 Images

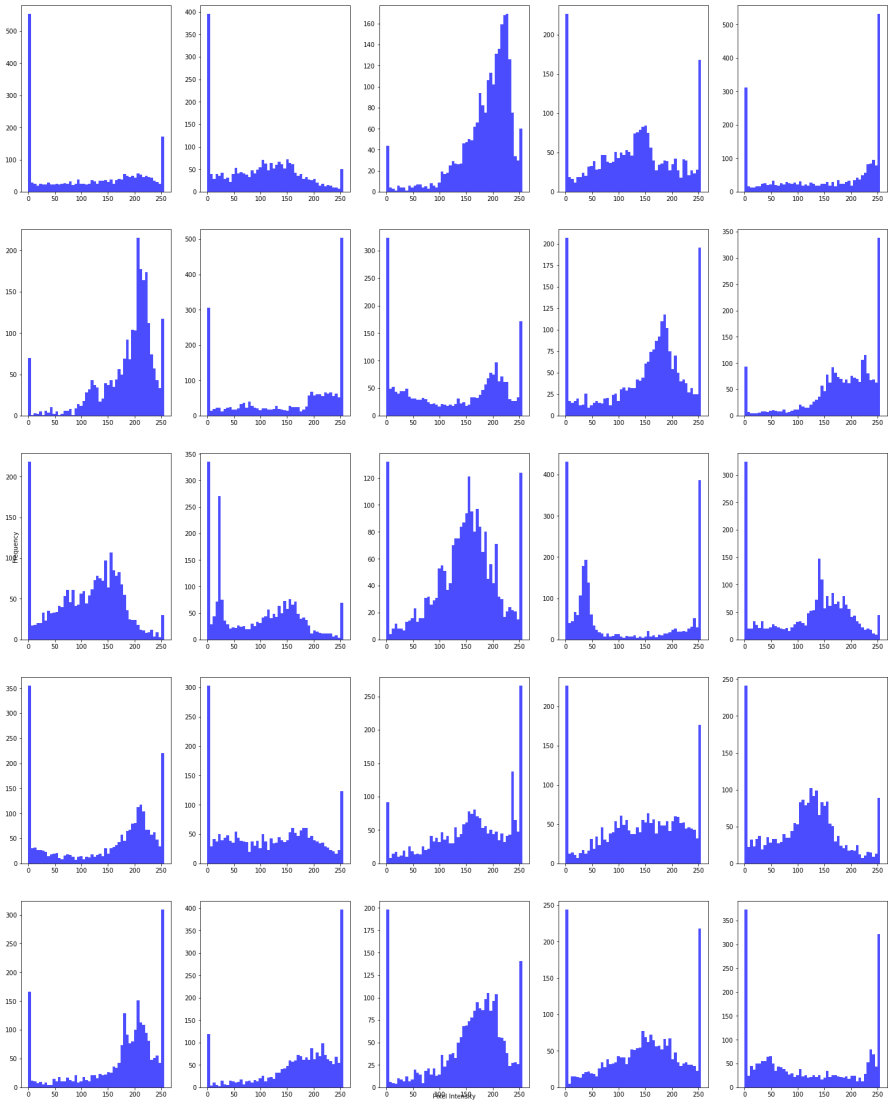


Figure 9: Pixel Intensity Distribution of 25 Images

5 CNN Model Architecture

After extensive analysis and research to identify the most suitable Convolutional Neural Network (CNN) architecture, we have narrowed it down to three variants. Among these three variants, the "Main Model" has demonstrated superior performance compared to the other two alternatives. In the following sections, we will provide a detailed explanation of our findings.

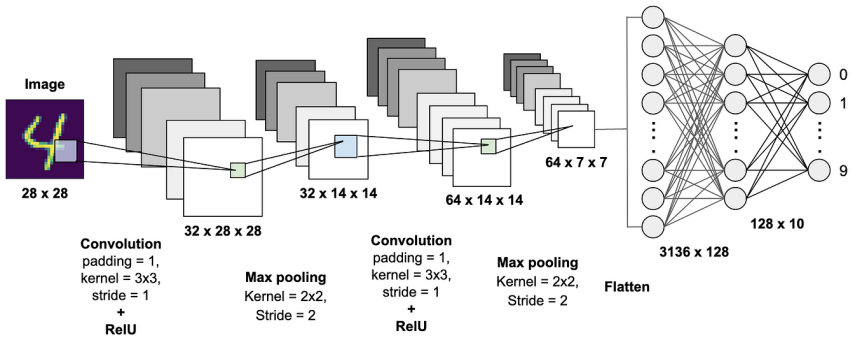


Figure 10: CNN Architecture Image [14]

The three CNN architecture variants under consideration are as follows:

1. Main Model (Variant 2)
2. Variant 1
3. Variant 3

5.1 Model Architecture

Model Variant 1

The Model Variant 1 architecture, designed for facial detection using Convolutional Neural Networks (CNNs), is detailed below:

Architecture Components

The Model Variant 1 architecture consists of convolutional layers, residual blocks, and fully connected layers. The key components of the architecture are as follows:

Convolutional Layers: The first convolutional layer (`firstConv`) processes the input image with 16 filters of size 3x3 and applies the Rectified Linear Unit (ReLU) activation

function. Subsequent layers (`secondConv`, `thirdConv`, and `fourthConv`) follow a similar pattern, increasing the number of filters gradually to capture more complex features.

Pooling Layers: Max pooling layers (`MaxPool2d`) are utilized to downsample the spatial dimensions of the feature maps, reducing computational complexity and enhancing translation invariance.

Residual Blocks: Two residual blocks (`firstResidual` and `secondResidual`) are integrated into the architecture. These blocks allow for the creation of deeper networks while mitigating the vanishing gradient problem, enhancing the model's ability to capture intricate facial features. [5]

Fully Connected Layers: The classifier section consists of fully connected layers (`Linear`) that further process the extracted features. These layers are responsible for mapping the high-dimensional feature space to the number of output classes, facilitating facial emotion detection.

Why This Specific Variant?

1. Residual Connections
2. Compact Classifier

We thought model variant 1 presents a promising architecture for facial detection tasks. Its feature hierarchy, inclusion of residual connections, and efficient classifier make it a viable candidate for our project. This was the reason we selected this model as one of the candidate models.

Model Variant 3

The Model Variant 3 architecture, designed for facial detection using Convolutional Neural Networks (CNNs), is detailed below:

Architecture Components

The Model Variant 3 architecture consists of convolutional layers, batch normalization, and fully connected layers. The key components of the architecture are as follows:

Convolutional Layers: The first convolutional layer (`primaryConv`) processes the input image with 10 filters of size 3x3, applying the Rectified Linear Unit (ReLU) activation function.

Batch Normalization: Following the second convolutional layer (`secondaryConv`), batch normalization is applied to normalize the activations and improve training stability.

Pooling Layers: Max pooling layers (`MaxPool2d`) are utilized to downsample the spatial dimensions of the feature maps, reducing computational complexity and enhancing translation invariance.

Fully Connected Layers: Two fully connected layers (`denseLayer1` and `finalDenseLayer`) process the extracted features, with the final layer mapping to the number of output classes.

Why This Specific Variant?

1. Batch Normalization for Stability
2. Pooling for Dimension Reduction

We selected Model Variant 3 due to its simplicity and efficiency. The use of batch normalization enhances training stability, and the architecture is well-suited for facial emotion detection tasks.

Model Variant 2

The Model Variant 2 architecture is a comprehensive deep learning model using Convolutional Neural Networks (CNNs) to perform image classification tasks with high accuracy. The network is structured to progressively refine features and enhance classification performance.

Architecture Components

The architecture is composed of several layers that work together to process images and make predictions:

Convolutional Layers: The network begins with `cnn1`, a layer that applies 64 filters to the input image. These filters are like little windows that look for patterns, such as edges and textures. As we move deeper (`cnn2` to `cnn13`), the number of filters increases, allowing the model to recognize more complex features like shapes or specific parts of objects. Each convolutional layer is paired with an activation function called ReLU, which helps the model learn non-linear patterns.

Batch Normalization: After each set of filters, we normalize the data using batch normalization. This process helps speed up learning and leads to faster training times [6]. It's like making sure the data doesn't have any extreme values that could throw off

the learning process.

Pooling Layers: Max pooling is used several times in the network (`MaxPool2d`). This technique reduces the size of the feature maps by picking the most prominent features and ignoring the less important ones. It simplifies the image without losing the core information, which makes the model faster and more efficient.

Fully Connected Layers: Toward the end of the network, we have three layers (`fc1`, `fc2`, and `fc3`) that take all the features we've found and use them to make a decision about what the image shows. The first two have a lot of neurons (4096 each) to combine all the different features the filters found, and they include a dropout of 0.5, which means we randomly ignore some neurons during training to prevent the model from becoming too dependent on any one part of the data. The last layer, `fc3`, tells us the classification of the image.

Why This Specific Variant?

1. Detailed Feature Learning
2. Efficient Training

This CNN architecture is a combination of current best practices and our extensive research of finding the best cnn for the project. It leverages a deep stack of convolutional layers for detailed feature extraction, batch normalization for accelerated learning, and dropout for robustness against overfitting [10].

5.2 Model Training Process

The training methodology for the selected CNN architecture (Model Variant 2) involved careful consideration of various hyperparameters and optimization strategies. The goal was to encourage the reduction of validation loss, prioritizing generalization over fitting to the training data. The key components of the training process are detailed below:

Hyperparameters Configuration

- **Learning Rate (LR):** LR was set to 0.001. A moderate LR is chosen to ensure stable convergence without overshooting the optimal weights during training.
- **Loss Function:** Categorical Cross-Entropy was employed as the loss function. This choice is suitable for multi-class classification tasks, aligning with the goal of facial detection [4].
- **Optimizer:** The Adam optimizer was selected. Adam adapts the learning rates for each parameter during training, providing an effective and efficient optimization strategy. Its adaptive nature makes it robust to various types of data and architectures.

- **Batch Size:** A batch size of 64 was utilized. Mini-batch training strikes a balance between the efficiency of stochastic gradient descent and the computational demands of processing the entire dataset in each iteration.
- **Epochs:** The model underwent training for 100 epochs. This choice allowed for a sufficient number of passes through the dataset to capture complex patterns in the data.
- **Early Stopping:** Early stopping was implemented as a regularization strategy. The training process was halted if the validation loss did not decrease for a consecutive number of epochs, defined by the tolerance parameter [1]. In this case, the tolerance was set to 5 epochs, and the minimum delta for considering improvement was 0.005.

Training Process

The model was trained using the specified hyperparameters and optimization strategies. The Adam optimizer adapted the learning rates for each parameter, facilitating convergence. The choice of Categorical Cross-Entropy as the loss function ensured effective optimization for multi-class classification.

The training process involved iterating through the dataset in mini-batches of size 64, updating the model parameters based on the computed gradients, and evaluating the model's performance on both the training and validation sets. Early stopping criteria were employed to prevent overfitting and ensure that the model generalized well to unseen data.

The emphasis during training was placed on minimizing the validation loss, aligning with the goal of achieving high performance on new, unseen facial images. This approach prioritizes the model's ability to accurately classify facial features without overfitting to the training data.

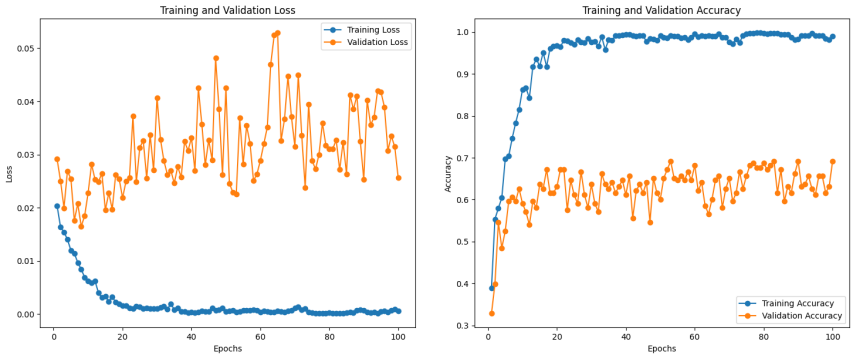


Figure 11: Main Model(V2) Training and Validation Loss / Accuracy

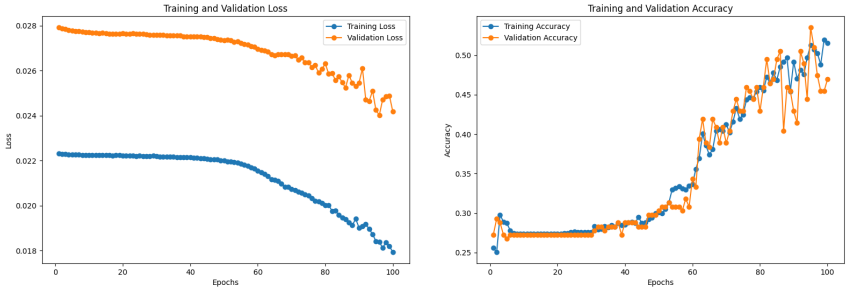


Figure 12: Variant 1 Training and Validation Loss / Accuracy

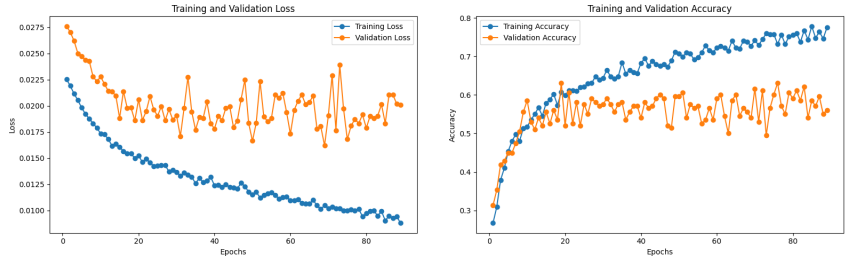


Figure 13: Variant 3 Training and Validation Loss / Accuracy

6 Model Evaluation

6.1 Performance Matrix

In the domain of facial image analysis using convolutional neural networks (CNNs), the performance metrics of precision, recall, and F-score hold substantial importance for applications such as facial recognition, emotion detection, and security systems.

Model	Macro			Micro			Accuracy
	P	R	F	P	R	F	
Main Model(V2)	0.72	0.70	0.70	0.70	0.70	0.70	0.70
Variant 1	0.54	0.53	0.53	0.53	0.53	0.53	0.53
Variant 3	0.64	0.64	0.63	0.65	0.65	0.64	0.65

Table 1: Performance Metrics of All Variants

Comparative Insights

Model Variant 1

- Shows significantly lower precision and recall, leading to higher instances of both false negatives and false positives.
- This model may be less suitable for critical applications, given its propensity for misidentification and missed detections.

Model Variant 2 (Main Model)

- Offers a balanced structure with parameterized activation functions and dropout. Incorporates batch normalization for improved generalization.
- Efficient in training and inference. High precision and recall, reducing overfitting, making it ideal for critical applications.

Model Variant 3

- A more compact design with fewer channels in convolutional layers. Includes batch normalization and pooling for efficiency.
- Faster in training and inference but may have lower accuracy in complex feature extraction.
- Demonstrates moderately high metrics but does not match the Main Model's performance, suggesting fewer errors than Variant 1 but not as few as the Main Model.

Specific Metric Implications

For facial image analysis:

- A model with *higher recall but lower precision* might rarely miss a face but could misidentify individuals, which is sub-optimal for security purposes but may be adequate for broad non security applications metrics.
- A model with *high precision but lower recall* would be preferable in scenarios where correct identification is paramount to avoid privacy or safety implications.

The Main Model's balanced metric profile suggests a superior capability to generalize from training data to new, unseen images, which is crucial for real-world application where variability is the norm. This balance is critical to ensuring the versatility and reliability of the model in various facial analysis contexts.

6.2 Confusion Matrix Analysis

The confusion matrix for the Main Model in the facial expression detection project provides valuable insights into the model's performance across different classes. We analyze the numbers carefully to identify which classes were most frequently confused, discuss potential reasons behind these misclassifications, and highlight well-recognized classes along with speculative reasons behind their success.

Most Frequently Confused Classes

- **Angry and Neutral:** The model confused 'Neutral' with 'Angry' on 17 occasions, the highest off-diagonal value in the matrix, indicating a notable misclassification rate.
- **Neutral and Focused:** There were 13 instances of 'Neutral' being misclassified as 'Focused', suggesting the model struggles to differentiate between these expressions.

Reasons Behind Misclassifications

- **Feature Representation:** The model may extract features that are not distinctive enough to differentiate between similar expressions.
- **Insufficient Training Data:** A lack of varied examples for certain expressions may prevent the model from learning to recognize them accurately.

Well-Recognized Classes

- **Focused and Bored:** Both 'Focused' and 'Bored' expressions are recognized with high accuracy, as indicated by 51 and 45 correct classifications respectively.
- **Angry:** The 'Angry' class is also well-recognized, with 35 correct predictions, showing the model's capability to identify this expression reliably.

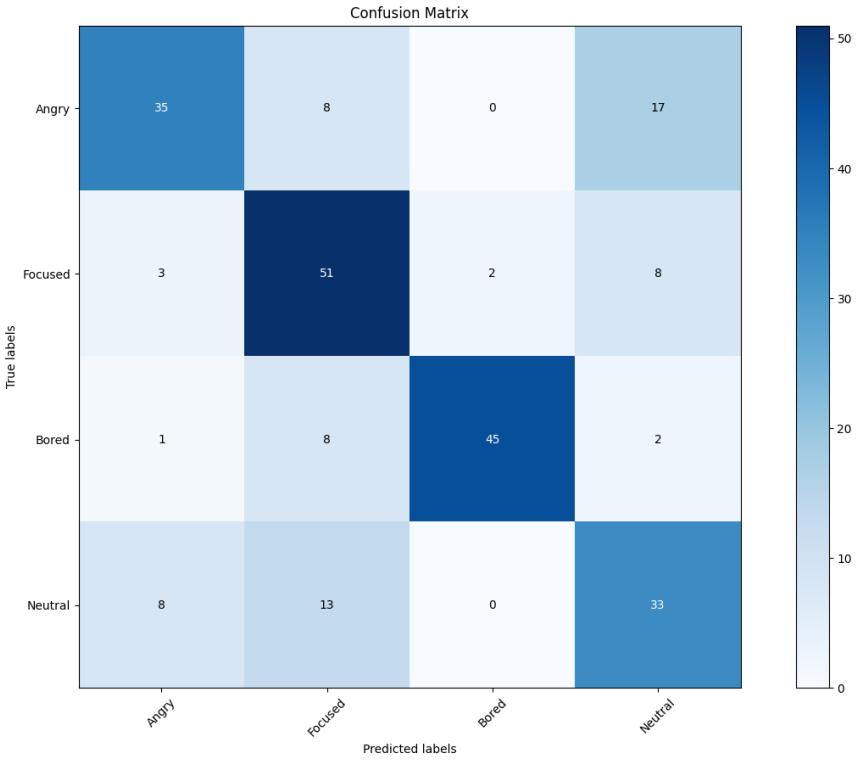


Figure 14: Main Model's Confusion Matrix

Reasons Behind Successful Recognition

- **Distinctive Features:** The expressions for 'Focused' and 'Bored' may possess unique features that are effectively captured by the model.
- **Effective Feature Learning:** The model may have successfully learned the critical features that distinguish 'Focused' and 'Bored' from other expressions.
- **Quality of Training Data:** High-quality and diverse training data for 'Focused' and 'Bored' expressions could contribute to the model's accurate recognition.

To further enhance the model's performance, one might consider increasing the diversity of the training data, particularly for underrepresented classes, and utilizing data augmentation to add variability to the training examples. Fine-tuning the model's architecture or hyperparameters may also help capture the subtleties between similar expressions more effectively.

6.3 Impact of Architectural Variations

In this section, we evaluate the performance of four different Convolutional Neural Network (CNN) architectures by slightly changing its filter's size and depth while keeping

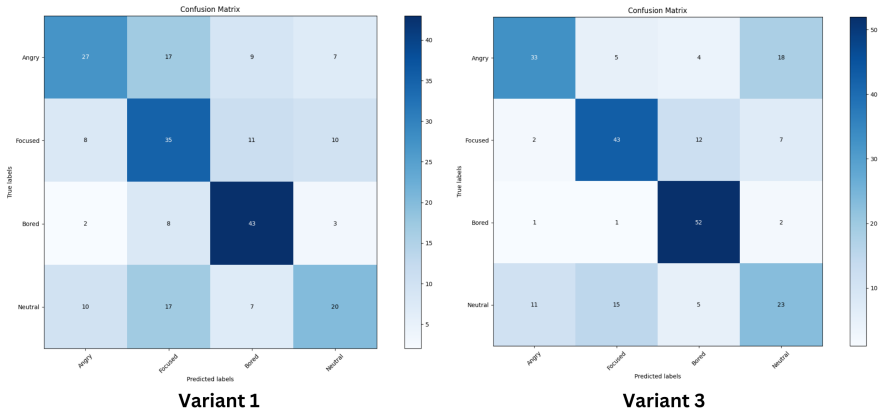


Figure 15: Confusion Matrix for Other Variants

the overall model architecture as it is. Each model's performance is assessed using several metrics: accuracy, precision, recall, and F1-score, both on macro and micro levels.

The first model, CNN-1, consists of 7 layers with (5, 5) filters. Despite its depth, it achieved the lowest accuracy of 0.54 and F1-scores of 0.52 (macro) and 0.54 (micro). The smaller filter size may have hindered the model's ability to capture sufficient relevant features. Additionally, the depth could introduce training complexities such as vanishing gradients, suggesting that more layers do not necessarily lead to improved performance.

Model	**Layers**	**Filter Size**	**Activation**	**Pooling**	**Optimiser**
CNN-1	7	(5, 5)	ReLU	MaxPool	Adam
CNN-2	6	(7, 7)	ReLU	MaxPool	Adam
CNN-3	6	(9, 9)	ReLU	MaxPool	Adam
CNN-4	5	(13, 13)	ReLU	MaxPool	Adam

Table 2: CNN Configurations: Exploring Depth and Filter Sizes

CNN-2, with 6 layers and (7, 7) filters, showed a significant improvement, yielding the highest accuracy and F1-scores of all models, both at 0.64. This indicates that a moderate increase in filter size can enhance feature capture without the drawbacks of excessive depth. The balance between the number of layers and filter dimensions seems to be optimal in this architecture, as it allows the network to learn more generalizable features.

With CNN-3, the model maintains 6 layers but expands the filter size to (9, 9). There is a slight decrease in performance metrics with an accuracy of 0.60 and a macro F1-score of 0.59. This suggests that larger filters may start to miss finer details critical for class distinction, despite capturing more global information.

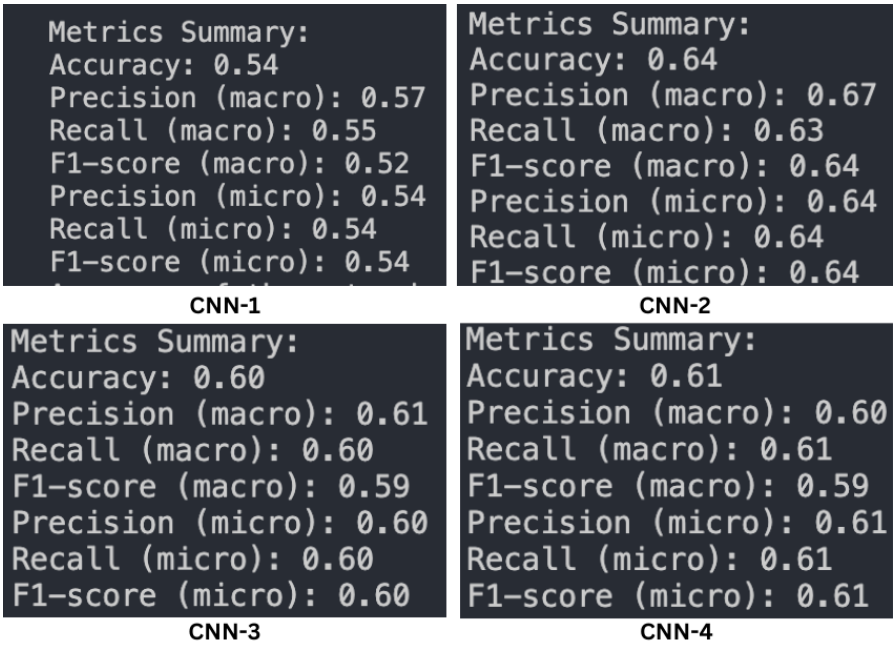


Figure 16: Experiments Results

Lastly, CNN-4's architecture with 5 layers and the largest (13, 13) filters, achieves an accuracy of 0.61 and macro F1-score of 0.59, comparable to CNN-3. The performance suggests that while the large filter size aims to capture global features, it risks overlooking detailed local features. However, the reduction in layers likely simplifies the training process, which can help mitigate some of the performance losses associated with larger filter sizes.

In conclusion, our evaluation demonstrates that a careful balance between the depth and filter size of a CNN is crucial for optimal performance. CNN-2 emerges as the best-performing model, suggesting that its configuration most effectively captures both global and local features without the model becoming overly complex.

7 Conclusions and Forward Look

Primary Findings

The Main Model, identified as Model Variant 2, has shown promising performance in facial expression recognition. With an accuracy of 70%, and precision, recall, and F1-scores all above 0.70 in macro average, the model exhibits a strong ability to generalize across various facial expressions. This balance of metrics indicates not only that the model is accurate on average but also that it does not overly favor any particular class, which is essential for fair and unbiased facial expression recognition.

The convolutional layers effectively extract spatial hierarchies of features from the facial images, while the max-pooling layers reduce dimensionality, helping to prevent overfitting and improving computational efficiency. The subsequent fully connected layers, coupled with dropout, further contribute to the model's generalization by preventing complex co-adaptations on training data. These architectural choices are likely key contributors to the model's success.

Given the complexity of human facial expressions and their subtle variations, the model's ability to discern these with a considerable degree of accuracy is noteworthy. However, there are opportunities to refine the model further, particularly in improving its ability to differentiate expressions that are inherently similar or where the dataset may not provide enough variance for robust learning.

Suggestions for Future Refinements

To further enhance the model's capabilities, the following refinements could be beneficial:

- Adjust the network depth or width to optimize model capacity.
- Experiment with advanced regularization or alternative activation functions to improve learning dynamics.
- Employ systematic hyperparameter tuning to find optimal training settings.
- Expand the training dataset, especially with more nuanced expressions, to aid the model in learning finer distinctions.

Such refinements could lead to significant improvements in performance, paving the way for a model that is not only highly accurate but also versatile and reliable across diverse facial expression recognition tasks.

Part 3 content Under the dataset

Mention the changes that we have done to dataset. That is, detail any additions, removals, or modifications to the dataset specifically.

Under Labeling

• Describe how your dataset was labeled for the two chosen bias attributes. • Describe how you verified the labels.

Under DataVis Update your three visualizations from Part I for the final dataset used in the project.

Under Evaluation Confusion matrix: Add the confusion matrix for your final model from Part III

New Section K-Fold cross validation

– Add the two tables showing the results from the 10-fold cross-validation (one table for your model from Part II, one for the final model), using the template shown in Table 1.

Discuss any significant observations or trends noted across the different folds, particularly in relation to the consistency of the model's performance.

steps to execute

1. evaluate the current model using k fold and fill up the table. 2. label the existing data Using age and gender. 3. look for the imbalance, visualise it and 4. make it balanced by augmentation finding new images. 5. In this part of the project, your challenge is to ensure that your AI system is not only technically proficient but also ethically sound. You are required to analyze your AI for two of the following categories: age, gender. For example, using suitably annotated testing data, investigate whether your system's performance across the four classes remains consistent across different genders. Experiment with methods such as re-balancing or augmenting your training dataset to address any identified biases. Your evaluation should encompass the network's performance on both the complete dataset and the subsets representing the chosen biased attributes. Evaluation: K-fold Cross-validation. Building upon the basic train/test split methodology from Part II of project, this phase requires you to employ k-fold cross-validation to enhance the robustness and reliability of your evaluation, especially across different classes. You are instructed to perform a 10-fold cross-validation (with random shuffling) on your AI model (the final model from Part II). This method involves dividing your dataset into 10 equal parts, using each part once as a test set while the remaining nine parts serve as the training set. This process is repeated 10 times, each time with a different part as the test set. This technique helps in assessing the model's performance more comprehensively and reduces the bias that can occur in a single train/test split. Utilize the functionalities provided by scikit-learn or skorch for implementing the k-fold cross-validation.6 It is important to avoid a manual, static split of the dataset for this process. Document the results of your 10-fold cross-validation in the project report, using the format shown in Table 1. This includes the performance metrics (accuracy, precision, recall, F1-score) for each fold, as well as the average across all 10 folds. In your report (see below), include a summary that discusses the overall performance consistency and any notable variations observed across different folds.

Table 1: How to report the performance metrics for each fold in the 10-fold cross-validation, together with the average over all folds. Use this table format for (i) your model from Part II of the project and (ii) your final, updated model from Part III. Detecting and Mitigating Bias. This critical phase of your project involves analyzing your AI model for potential biases and implementing strategies to mitigate them. The process is divided into three main steps: data collection, analysis for bias, and retraining the system if needed. Start from the model you developed and saved in Part II of the project for the initial bias evaluation.

Table 2: Reporting the result from the bias analysis in this table. For your two selected bias attributes (here "age" and "gender" as an example), report the macro-averaged metrics as shown above. Use this table layout for both (i) your model from Part II and (ii) your updated, final model from Part III. Note: you may have different groups for each attribute (e.g., more age groups), depending on your dataset. Data Collection for Bias Analysis: Segment your dataset based on the chosen attributes (e.g., age, gender, race). Ensure that these attributes are accurately annotated in your dataset, either through existing labels or through additional annotation efforts as part of this project. Evaluate the saved model from Part II on each group separately using a standard train/test split (like you did in Part II), collecting macro-averaged performance metrics (Accuracy, Precision, Recall, F1-Score) for each demographic group. Fill these metrics into the Bias Analysis Table (Table 2). Repeat for your second chosen attribute. Analysis of Collected Data: Analyze the filled Bias Analysis Table to identify any significant performance discrepancies across different demographic groups, which may indicate biases. Pay close attention to variations in performance metrics among the groups within each attribute. Retraining and Mitigation: If biases are detected, consider dataset augmentation to enhance the representation of underrepresented groups. This might include sourcing additional images that accurately represent these groups, exploring different lighting conditions, backgrounds, or responsibly using synthetic data generation techniques to diversify the training data.7 After augmenting your dataset, retrain your model and reassess its performance using the bias analysis approach. Report the results in a second Bias Analysis Table for the new model. Document any performance changes and the steps taken to mitigate bias in your report. Ensure that all steps, findings, and changes in model performance after mitigation are thoroughly documented in your project report (see below). This detailed analysis is crucial for demonstrating your understanding of ethical AI development and the importance of creating fairer AI systems.

References

[1] Yingbin Bai et al. *Understanding and Improving Early Stopping for Learning with Noisy Labels*. 2021. arXiv: `2106.15853 [cs.LG]`.

[2] Pierre-Luc Carrier and Aaron Courville. *Facial Expression Recognition 2013 Dataset*. URL: `https://www.kaggle.com/datasets/msambare/fer2013`.

[3] Google. *Google Images*. 2023. URL: https://www.google.com/imghp.

[4] Elliott Gordon-Rodriguez et al. *Uses and Abuses of the Cross-Entropy Loss: Case Studies in Modern Deep Learning*. 2020. arXiv: 2011.05231 [stat.ML].

[5] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[6] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[7] mbartenschlag. 2019. URL: https://github.com/mitre/biqt-face/commits/master/config/haarcascades/haarcascade_frontalface_alt2.xml.

[8] Microsoft. *Bing Images*. 2023. URL: https://www.bing.com/images.

[9] nikhilaggarwal3. *Data Visualization with Python*. 2022. URL: https://www.geeksforgeeks.org/data-visualization-with-python/.

[10] Keiron O'Shea and Ryan Nash. *An Introduction to Convolutional Neural Networks*. 2015. arXiv: 1511.08458 [cs.NE].

[11] OpenAI. *ChatGPT by OpenAI*. 2023. URL: https://openai.com/blog/chatgpt.

[12] Seth Shostak. *Sharpening Basics: Here's How Unsharp Masking Makes Your Photos Look Extra Crispy*. 2017. URL: https://www.shutterbug.com/content/sharpening-basics-here%E2%80%99s-how-unsharp-masking-makes-your-photos-look-extra-crispy.

[13] Shahid Akhtar Khan. *How to crop and save the detected faces in OpenCV Python?* 2022. URL: https://www.tutorialspoint.com/how-to-crop-and-save-the-detected-faces-in-opencv-python.

[14] Shreyak. *CNN Image*. 2020. URL: https://becominghuman.ai/building-a-convolutional-neural-network-cnn-model-for-image-classification-116f77a7a236.

[15] The Matplotlib development team. *Matplotlib 3.8.0 documentation*. URL: https://matplotlib.org/stable/.

[16] WFDownloader. *Bulk Image Downlaoder*. 2023. URL: https://www.wfdownloader.xyz/.