Abstract: In the era of the Fourth Industrial Revolution, machine learning has emerged as a critical technology that is being used in a wide range of real-world applications, ranging from Natural Language Processing to Computer Vision due to its ability to analyse and extract insights from vast amounts of data. This report focuses on two applied machine learning tasks: spam detection in text and facial landmark alignment in images. For Task 1, two classification models namely Naïve Bayes and Logistic Regression, were developed, evaluated, and compared in terms of performance. Task 2 involved experimenting with various models before adopting a hybrid cascaded regression framework leveraging SIFT-based features to iteratively refine landmark predictions. The report outlines the design choices behind each approach, justifies the selected methodologies, and presents a thorough evaluation of results on the respective validation datasets.

Splitting of Labelled Dataset

To evaluate and compare the models objectively, the labelled dataset was split into a training set (80%) and a validation set (20%) for all models used in both of these tasks.

Task 1 – Spam Detection

Introduction

Spam Detection is one of the most ubiquitous and impactful applications of natural language processing (NLP), aiming to classify text into two categories – 'Spam' or 'Not Spam' by identifying and filtering out unsolicited or malicious messages. It holds substantial practical relevance in domains like online content moderation and filtering of emails and messages where automated detection of fraudulent or harmful content is critical. This process typically relies on sophisticated pattern recognition and anomaly detection techniques which are a hallmark of modern NLP (Ahmad, 2024).

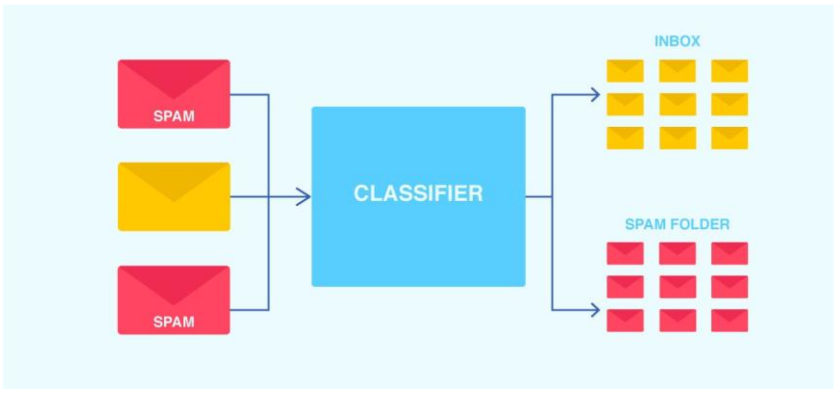


Figure 1: Classification of Text Data

Pre-Processing

Before any model training, the raw text data needs to be systematically cleaned and structured for effective classification. Raw text often contains irrelevant or extraneous information that can hinder model performance, , if not properly addressed.

Lowercasing

The first step in preprocessing involved converting all characters in the text to lowercase. This prevents feature duplication by ensuring that identical words with different capitalizations are treated as the same token. As a result, the vocabulary size was significantly reduced, enabling the model to focus on semantic content rather than case sensitivity.

Example:
Before – ANNOUNCEMENTS: Better Hurry! The organisers are finalising the arrangements & distributing flyers to attendees by 4:00pm.
After – announcements: better hurry! the organisers are finalising the arrangements & distributing flyers to attendees by 4:00pm.

Removal of Punctuation, Digits and Special Symbols

Using regular expressions, all non-alphabetic characters including punctuation marks, special characters and numerical digits were removed. This helps in eliminating irrelevant noise, as such elements are uninformative for spam classification and would otherwise unnecessarily expand the feature space.

Continuing example from above:
Before – announcements: the organisers are finalising the arrangements & distributing flyers to attendees by 4:00pm.
After – announcements better hurry the organisers are finalising the arrangements distributing flyers to attendees by pm

Tokenisation

The Natural Language Toolkit (NLTK) library was used to tokenize the text, breaking it down into individual words (or tokens) that represent meaningful and easily analysable components. These tokens would serve as the foundation for subsequent transformations and feature extraction.

Continuing example from above:
Before – announcements better hurry the organisers are finalising the arrangements distributing flyers to attendees by pm
After – ['announcements', 'better', 'hurry', 'the', 'organisers', 'are', 'finalising', 'the', 'arrangements', 'distributing', 'flyers', 'to', 'attendees', 'by', 'pm']

Stopword Removal

Common English stopwords (e.g. 'the', 'it', 'and') were removed using NLTK's built-in stopword list. These high-frequency words are usually articles, prepositions, conjunctions that contribute little semantic value in the context of text analysis and only add unnecessary noise. Removing them reduces the vocabulary size, improve the signal-to-noise ratio, and eliminates low-discriminative tokens, thereby enhancing the efficiency and accuracy of classification.

Continuing example from above:
Before – ['announcements', 'better', 'hurry', 'the', 'organisers', 'are', 'finalising', 'the', 'arrangements', 'distributing', 'flyers', 'to', 'attendees', 'by', 'pm']
After – ['announcements', 'better', 'hurry', 'organisers', 'finalising', 'arrangements', 'distributing', 'flyers', 'attendees', 'pm']

Stemming

The final preprocessing step was stemming, a text normalization technique that reduces words down to their root form, also known as a "lemma" in linguistics (Murel and Kavlakoglu, 2023). Stemming reduces the vocabulary size by grouping semantically similar words under the same representation and unifying word forms. This helps the model generalise better because it allows it to recognize that words with the same root often have similar meanings, even if they appear in different forms in the training and test data. The well-known and widely used Porter Stemmer was employed for this task.

Continuing Example from above:

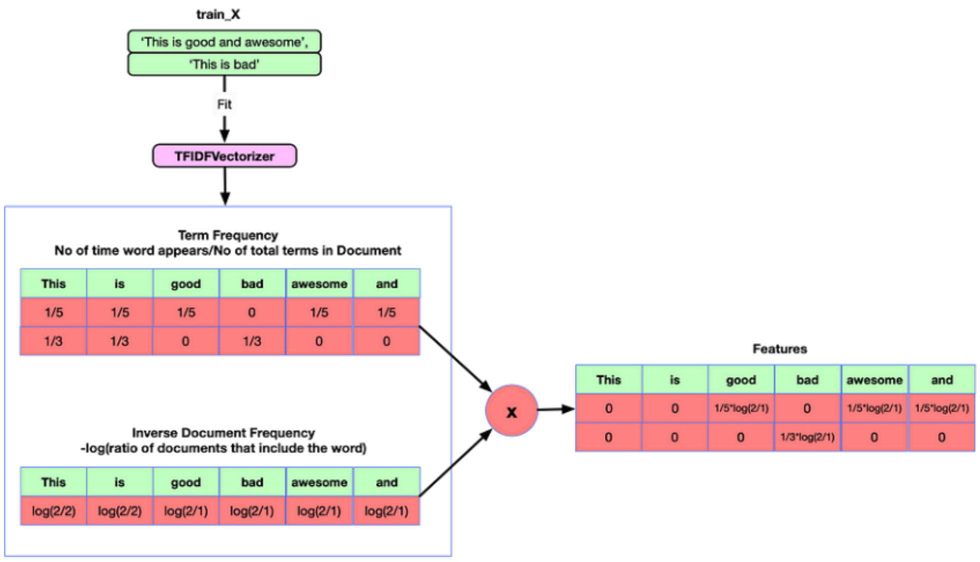
Before – ['announcements', 'better', 'hurry', 'organisers', 'finalising', 'arrangements', 'distributing', 'flyers', 'attendees', 'pm']
After – ['announc', 'better', 'hurri', 'organis', 'finalis', 'arrang', 'distribut', 'fli', 'attende', 'pm']

Feature Extraction

After the raw text data had undergone all preprocessing steps, it needed to be transformed into a numerical representation that would be suitable for input into machine learning models. This process is known as text vectorisation and can be achieved using various tools and techniques.

TF-IDF Vectorizer

Term Frequency–Inverse Document Frequency (TF-IDF) is a statistical technique used to the importance of a word within a document relative to a corpus. It combines Term Frequency (TF), which measures how often a term appears in a document, and Inverse Document Frequency (IDF), which evaluates the uniqueness of the term across all documents. While TF treats all terms as equally significant, many high-frequency words (e.g., 'and', 'the') offer little discriminative value. TF-IDF mitigates this by down-weighting commonly occurring terms and amplifying rarer, more informative ones (Singh, 2022). This weighting helps the model to highlight contextually meaningful terms and suppress semantically ubiquitous and neutral ones, thus improving spam detection performance.



Term Frequency
No of time word appears/No of total terms in Document

This	is	good	bad	awesome	and
1/5	1/5	1/5	0	1/5	1/5
1/3	1/3	0	1/3	0	0

Inverse Document Frequency
-log(ratio of documents that include the word)

This	is	good	bad	awesome	and
log(2/2)	log(2/2)	log(2/1)	log(2/1)	log(2/1)	log(2/1)

Features

This	is	good	bad	awesome	and
0	0	1/5*log(2/1)	0	1/5*log(2/1)	1/5*log(2/1)
0	0	0	1/3*log(2/1)	0	0

Figure 2: TF-IDF Vectorizer

To reduce dimensionality and mitigate overfitting, the TF-IDF vectorizer was limited to the top 5,000 most frequently occurring terms in the dataset. Both unigrams (single words) and bigrams (pairs of adjacent words) were included as features in the vectorizer to capture more contextual information and relationships between words. For example, "not good" as a bigram carries a very different meaning than "good" as a unigram alone.

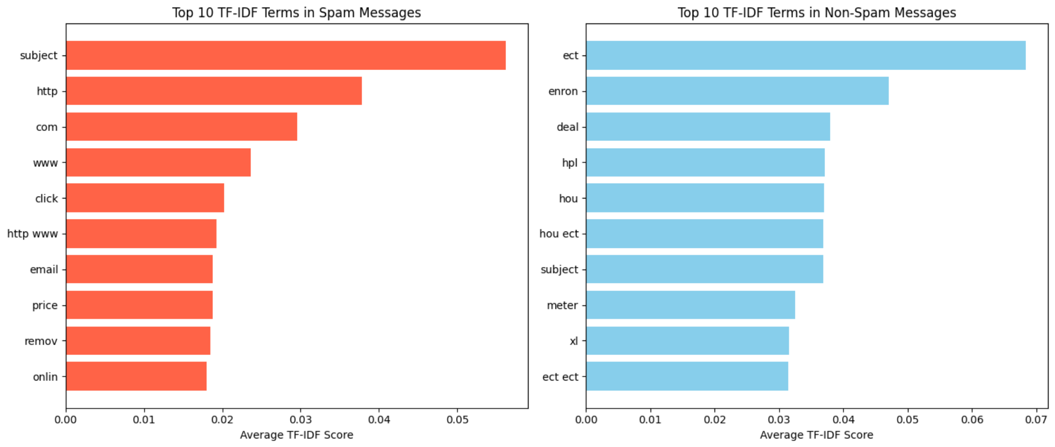


Figure 3: Highest scoring TF-IDF tokens in Spam and Non-Spam messages

Classification Models

Multinomial Naïve Bayes

Naïve Bayes is a probabilistic model that applies Bayes' theorem under the assumption that features (in this case, tokens or n-grams) are conditionally independent given the class label. In the context of text classification, this implies that the presence of one word in a document is assumed to be independent of the presence of any other word. Based on this assumption, Naïve Bayes estimates the probability of each class (spam or not spam) and assigns the class with the highest posterior probability, calculated using the observed word frequencies within the training data.

Advantages:

- Extremely fast to train and predict.

Disadvantages:

- Performance tends to degrade when important words occur in both classes with similar frequency due to strong independence assumption.

Logistic Regression

Logistic Regression is a discriminative linear classifier that models the probability of a binary outcome using a linear decision boundary to separate classes based on feature weights. It uses a sigmoid function to reduce any value to a range between 0 and 1. Unlike Naïve Bayes, it does not assume feature independence and can directly optimize classification performance through gradient descent.

Advantages:

- Generally, achieves higher accuracy and better decision boundaries than Naive Bayes.

Disadvantages:

- Slower to train on large datasets.

Comparison

Naïve Bayes

Metric	**Not Spam (0)**	**Spam(1)**	**Macro Avg**	**Weighted Avg**
Precision	0.99	0.88	0.93	0.95
Recall	0.94	0.97	0.96	0.95

F1-score	0.96	0.92	0.94	0.95
Support	509	215	724	724

The Naïve Bayes model achieved 480 true positives, 29 false positives, 208 true negatives and 7 false negatives.

Logistic Regression

Metric	**Not Spam (0)**	**Spam(1)**	**Macro Avg**	**Weighted Avg**
Precision	0.99	0.96	0.98	0.98
Recall	0.98	0.98	0.98	0.98
F1-score	0.99	0.97	0.98	0.98
Support	509	215	724	724

The Logistic Regression model achieved 501 true positives, only 8 false positives, 211 true negatives and only 4 false negatives.

Failure Cases

Although both models performed quite well, messages that have a high degree of text obfuscation (where words are intentionally scrambled to bypass detection) and extra nonsense words were sometimes misclassified as not containing spam. This highlights a limitation of techniques like TF-IDF which rely solely on surface-level n-gram.

Example:

Subject: re : i know what you wish for
goodbye
gargle copolymer hormoneextreme
monic puerilecafeteria nagging longfreethink
arise beachcombconserve dignitary alliancegemlike
clare boggingcrowberry hecate authenticatemorphology
courtney saxophonesummand

[True Label: Spam, Predicted: Not Spam]

Similarly, non-spam emails that contained typical spam-indicative patterns like URL mentions or "click on the link" phrases which commonly appear in phishing or promotional content were sometimes flagged incorrectly. This underscores another limitation of such models which is that they don't always understand broader context.

Example:

Subject: holiday party - save the date
please click on the link below
save the date

[True Label: Not Spam, Predicted: Spam]

Since the Logistic Regression model achieved better performance on the validation data across the board, with higher precision, recall and F1-scores; it was chosen as the final model to train the full training dataset on and was used to generate the predictions on the test data.

Task 2 – Facial Alignment

Face Alignment is a computer vision task that involves locating facial landmarks on a human face in images. Accurate detection of these key points is essential in various real-world applications like facial recognition, expression analysis and avatar animation.

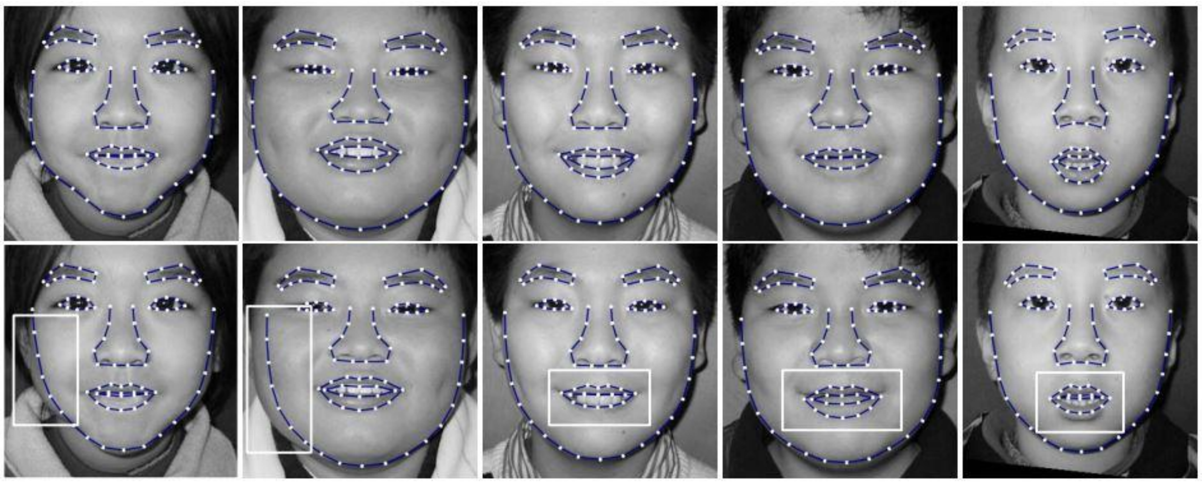


Figure 4: Detecting facial features

Pre-Processing

Prior to feature extraction, all facial images and their corresponding landmark annotations were pre-processed to standardize the input data and reduce complexity.

Grayscale Conversion

Since colour information is not essential for local shape or texture-based landmark detection, all images were converted to grayscale to reduce the input dimensionality.

Image Resizing and Normalisation

Each image was resized to a fixed resolution of 96*96 pixels. Resizing the images to smaller resolution reduces the chance of overfitting to specific fine-grained details that may be irrelevant and allows the model to focus more on the more essential facial features.

Pixel Normalisation

Pixel values in the image were normalized to a range between 0 and 1 by dividing them by 255. This improves gradiant stability during training and reduces sensitivity to lighting variations.

Landmark Scaling

Since the images were resized, their corresponding landmark coordinates also need to be scaled proportionally to match the new dimensions. Preserving the geometric relationship between the landmarks and facial features is essential and failing to do so would lead to systematic prediction errors.

Feature Extraction

The effectiveness of a face alignment model relies heavily on the quality and informativeness of the features extracted from each image. Instead of using raw pixel intensities which are sensitive to lighting conditions, noise, and misalignment, we employed Scale Invariant Feature Transform

(SIFT). SIFT effectively captures gradient patterns and edge orientation in local patches, making it well-suited for detecting consistent facial structures across diverse images.

Landmark-Based SIFT

Initially, SIFT descriptors were extracted at landmark locations during training. This technique proved relatively effective on the validation set which also had access to the landmark locations. However, since the images in the testing dataset do not have marked keypoints, SIFT had to be performed on the centred mean face shape for the test images. This discrepancy between the technique used during training and that used in testing caused the model to make extremely bad predictions on the test images, even if the validation error was pretty good. As a result, this model was completely ruled out.

Uniform Grid Pattern-Based SIFT

The next method that was tested was covering the entire image in a uniform grid and extracting features from each patch. This approach ensured a consistent and pose-invariant representation across all images, improving the model's generalisation to faces in varied orientations. However, this high dimensionality made the model prone to overfitting due to lack of semantic focus. In this model, features were being extracted from irrelevant places which added unnecessary noise and complexity.

5-Point Prior-Based SIFT

Instead of an evenly spaced grid across the whole image, this new method only extracted SIFT in a useful pattern. The average coordinate location for each landmark was calculated and feature extraction was only done in small patches around those points. This allowed the model to focus on only semantically meaningful locations and not get distracted by features in irrelevant places. The lower dimensionality also reduced the chance of overfitting and generalised more effectively across diverse face geometries and orientations.

Since the 5-point Prior-Based SIFT, gave the lowest validation error, that architecture was chosen for the final model.

Model Architecture

Regular Ridge Regression

In the "one-shot approach", a single ridge regression model was trained to predict the entire set of landmark coordinates in a single step, directly from the extracted SIFT features. In the training phase, the model learns a set of weights that map the extracted features to the corresponding shape updates, trying to minimise the regularised least square loss. Regularization helps prevent overfitting by penalizing large weight magnitudes, encouraging a simpler and more generalizable model. Once trained, the model performs predictions by applying the learned weights to the new set of features. Although this model was relatively simple to implement, it lacked precision and often predicted too close to the mean shape, failing to adapt well to facial variation.

Cascaded Regression

Cascaded regression is an iterative machine learning framework designed to improve predictions stage-by-stage. Instead of predicting the final output in one step, cascaded regression learns a series of progressively refined updates, allowing the model to correct its own errors over multiple stages. In the model we used, we made the patch sizes used to extract SIFT upon successively smaller after each cascade, going from a patch size of 16 to 12 to 8 to 6 to 4 to 3.

At each stage 't', the regressor learns to predict a delta shape update $\Delta S^{(t)}$ based on the current estimates of the landmarks S^t:

$$\Delta S^{(t+1)} = S^t + \Delta S^{(t)}$$

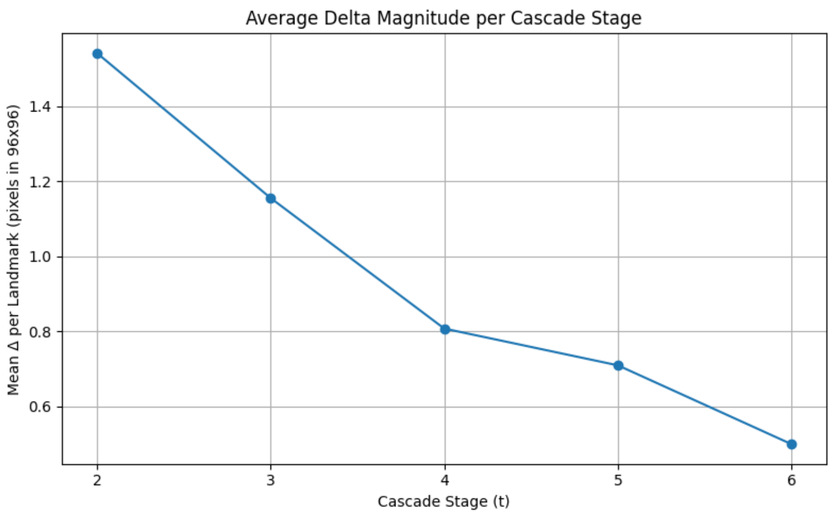


Figure 5: Change in delta magnitude across cascade stages

As shown in the figure above, the average amount of the landmark updates applied at each stage of regression continuously declines from over 1.5 pixels at stage 1 to under 0.5 pixels at stage 6 (this is in the reduced 96*96 resolution). This clearly indicates that the early regressors make large structural corrections, while the later stages fine-tune the shape with progressively smaller adjustments applied Since features are extracted relative to the current prediction, each stage uses increasingly accurate local context. This aligns with the intended design of cascaded regression models, where coarse-to-fine refinement improves both accuracy and robustness.

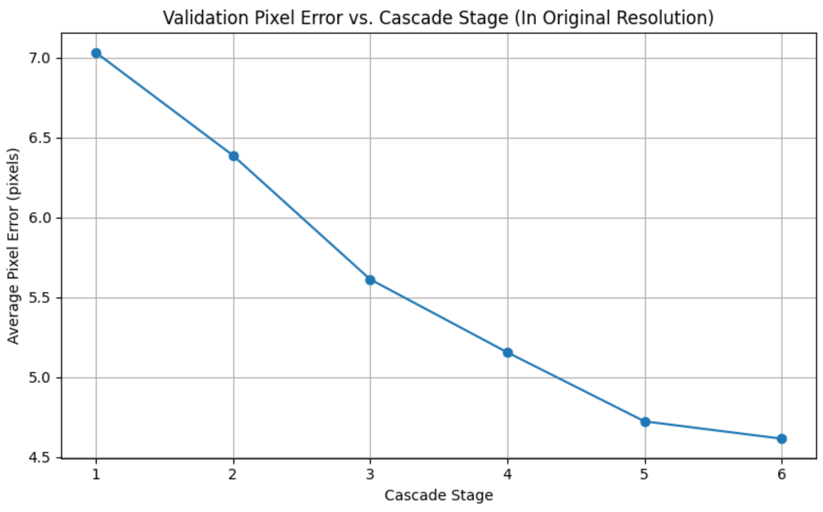


Figure 6: Change in validation error across cascade stages

Empirically, it was observed that average pixel error in the validation set decreased consistently as more stages were added until the 6th stage when error results began plateauing. This trend demonstrates that each stage contributes meaningfully to refining the landmark predictions, with diminishing error as the cascade progresses.

Comparison

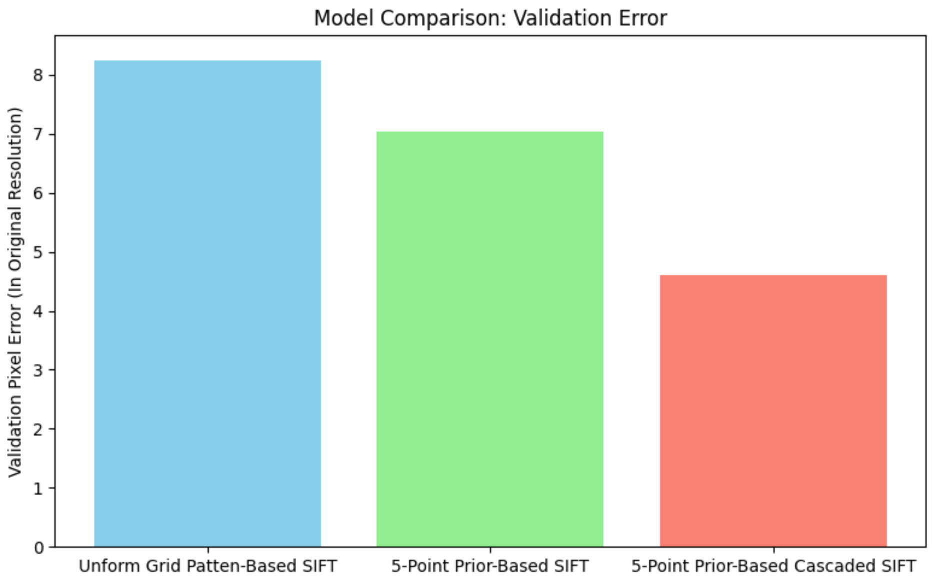


Figure 7: Comparison between different models

The uniform grid SIFT model, while covering the entire face area evenly, performed the worst due to high dimensionality and lack of semantic focus. It has a validation pixel error rate of 8.25. The 5-point prior-based one-shot regression model improved on this by only extracting features from semantically meaningful facial regions, reducing noise and overfitting. It had a validation pixel error rate of 7.03. However, the best performance was achieved by the 5-point prior-based cascaded regression model, which refined predictions iteratively across six stages with smaller patches applied at each stage. This model achieved the lowest average pixel error of 4.61 and was therefore selected as the final model for deployment and evaluation on the test set.

Qualitative inspection of the results revealed that the baseline models often failed to adapt to diverse face shapes, particularly around the jawline and eyebrows. They also struggled to predict the points accurately when the face was not oriented frontally or which contained a lot of noise near the landmark points (e.g., eye make-up, wearing glasses and wrinkles on face).In contrast, the cascaded regression model performed much better and was able to predict accurate results, even in semantically difficult images.

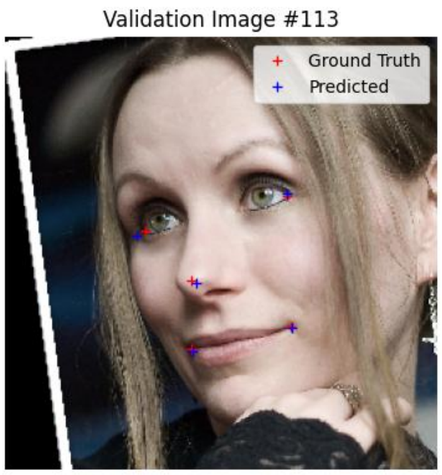


Figure 9: Predicted vs Actual Landmarks

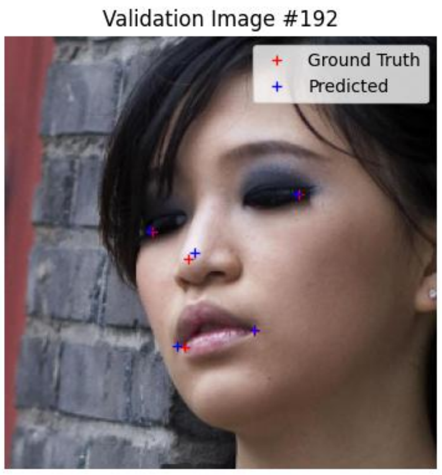


Figure 8: Predicted vs Actual Landmarks

Failure Cases

With some images, none of the models were able to predict all the landmark locations accurately. In figure 10, there is clear misplacement of the right eye and nose landmarks. The likely cause of this failure is that the subject's head is tiled causing an extremely asymmetrical facial expression. Images that deviate so much from the mean face shape are difficult to predict accurately. Figure 11 is yet another example of incorrect predictions. In this image, the model is unable to follow the curvature and rotation of the facial structure and ends up misaligning the mouth landmark points. Since the model is trained predominantly on upright, frontal faces and initializes predictions from the mean positions of the landmark features, these images are particularly difficult for the model to predict accurately.

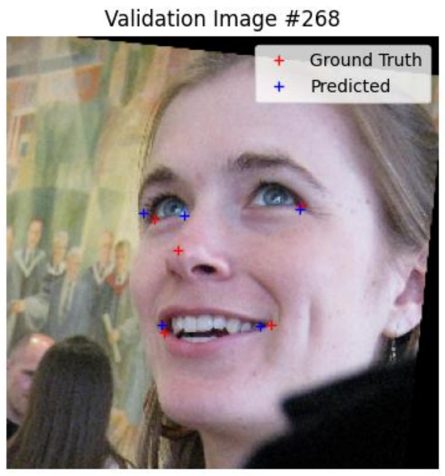


Figure 11: Failure Case 1

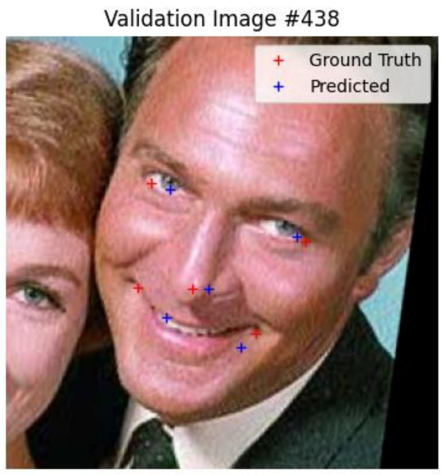


Figure 10: Failure Case 2

Since the 5-point prior-based cascaded regression model had the least validation error, it was chosen as the final model to train the full training dataset on and was used to generate the predictions on the test data.

Appendix

This link will take you to the Google Colab notebook where I wrote the code for Task 1.
This link will take you to the Google Colab notebook where I wrote the code for Task 2.

It is important to note that both of these python notebooks contain multiple models of which only 1 was selected for my final submission and used on the unlabelled test data. Furthermore, these notebooks were also downloaded and submitted as .ipynb files along with this report to canvas. In the submission, you will also find two csv files containing the predictions of my model for both tasks.

References

Ahmad, M.S. (2024). *Deep Learning 101: Lesson 25: Spam Detection with NLP*. [online] Medium. Available at: https://muneebsa.medium.com/deep-learning-101-lesson-25-spam-detection-with-nlp-91e0e57db9d5.

Liang, N.L., Wen, N.F., Xu, N.Y.-Q., Tang, N.X. and Shum, N.H.-Y. (2006). Accurate Face Alignment using Shape Constrained Markov Network. doi:https://doi.org/10.1109/cvpr.2006.45. (Used an image)

Murel, J. and Kavlakoglu, E. (2023). *What is stemming?* [online] Ibm.com. `
Available at: https://www.ibm.com/think/topics/stemming.

Singh, S. (2022). *CountVectorizer vs TfidfVectorizer*. [online] Medium.
Available at: https://medium.com/@shandeep92/countvectorizer-vs-tfidfvectorizer-cf62d0a54fa4.