Applied Machine Learning – Ruairi Bradley 279123 - Task 1:

Introduction

Spam, something everyone deals with and no one wants – unwanted or malicious e-mail/SMS. For this task I have built a model for classification of spam or not spam (ham) using text features and supervised learning. The journey started with a rule-based keyword selecting model, which I then advanced to a Naïve Bayes until finally settling on TF-IDF feature extraction, Logistic Regression, and text data augmentation (synonym replacement, random word deletion) as in testing it yielded the best and most robust results. I found pre-processing was key for a good output, the steps I used included: lowercasing, punctuation, stopword removal and lemmatization.

Through this iterative design the model managed to obtain great performance on the validation set, achieving approximately 1.00 accuracy, with 1.00 precision for ham and 0.99 precision / 1.00 recall for spam. Below I go into detail on the set up.

Methods

Overview

Firstly, in order to maximise results, I explored and tuned the pre-processing, feature representation and model choices.

The model utilised the training data (spam_detection_training_data.csv), which holds the text and labels for it they are spam or not, for development and validation. Additionally, the test data (spam_detection_test_data.csv), which was unlabelled text, was reserved for the final predictions.

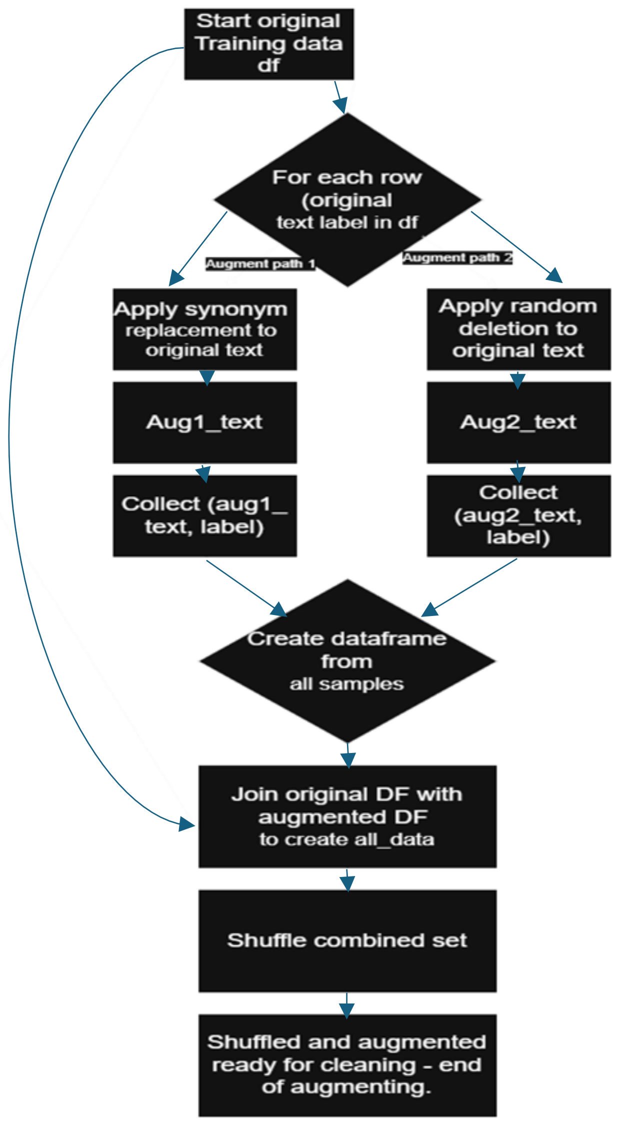
Text Preprocessing:

To ensure the feature extraction had good inputs, the text data was cleaned and standardised (clean_text function) with the following steps (Lecture 13):

1. Lowercasing: Ensures consistency (e.g., "Sell" and "sell" are identical)
2. Removing Non-Alphabetic Characters: Stripping punctuation and digits to focus on words (lowers noise).
3. Tokenization: Texts were split into word tokens using nltk.wordpunct_tokenize.
4. Stopword Removal: Common words (e.g., "the", "is") were removed using NLTK's standard list, as these high-frequency words typically offer little value.
5. Lemmatization: Words are reduced to their base form (e.g., "running" to "run") using WordNetLemmatizer. This standardises vocabulary by grouping and was preferred over stemming, to preserve more semantic context.

Augmenting The Dataset

To predict spam accurately, increasing the size of the data set was an obvious choice to improve model reliability. Two augmentation strategies were applied, essentially tripling the size of the training set – listed below:



Start original Training data df
For each row (original text label in df
Augment path 1
Augment path 2
Apply synonym replacement to original text
Apply random deletion to original text
Aug1_text
Aug2_text
Collect (aug1_text, label)
Collect (aug2_text, label)
Create dataframe from all samples
Join original DF with augmented DF to create all_data
Shuffle combined set
Shuffled and augmented ready for cleaning - end of augmenting.

Figure 1- Augmenting Data Pipeline: Each training message has two variants – one: two synonym replacements and two: with ~15 % random word deletion. Augmented samples are merged with the originals and the set is shuffled before cleaning and feature extraction.

1. Synonym Replacement: Up to two words (n=2) per message were randomly replaced with WordNet synonyms, to introduce diversity, which I believe deepens spam identification, despite varied wording (Lecture 18).

2. Random Deletion: Approximately 15% (p=0.15) of words were randomly removed, simulating informal or noisy text that might be common in SMS (text messages).

Feature Representation

TF-IDF Once cleaned, the augmented text was converted into numerical features using Term Frequency-Inverse Document Frequency (TF-IDF), this was because:

TF-IDF assigns weights to words based on their frequency in a document (TF) against their rarity across the entire corpus (IDF), placing importance on terms that occur frequently in specific messages but aren't frequent overall (Lecture 18).

I chose this over Bag-of-Words for its ability in highlighting discriminative terms for spam/ham classification (Lecture 13).

max_features set to = 3000 for the TfidfVectorizer as I found it balanced computational cost and overfitting.

Prediction Model

Logistic Regression as the task is binary classification by nature (spam/ham).

A Logistic Regression classifier was deployed with (sklearn.linear_model.LogisticRegression). It's an efficient linear model suitable for TF-IDF text data (Lecture 15). Also, it performed better than the Naïve Bayes experiment so was the desired way to move forward.

Key parameters choices & loss were:

1. C=5: I chose this inverse of regularization strength based on validation performance (Lecture 15).
2. solver='liblinear': It is suitable for binary classification and this dataset size.
3. max_iter = 1000: Ensured convergence.

4. The model implicitly optimizes for Log Loss (binary cross-entropy).
5. Training: The model was trained on an 80% split of the fully augmented, cleaned, and TF-IDF vectorized dataset then validated on the other 20% (Lecture 14).

Baselines for Comparison: Performance was compared to two baselines:

1. A Rule-Based Keyword Classifier (Lecture W07_Text_classification).

2. A Naive Bayes classifier with Bag-of-Words features (Lecture 15).

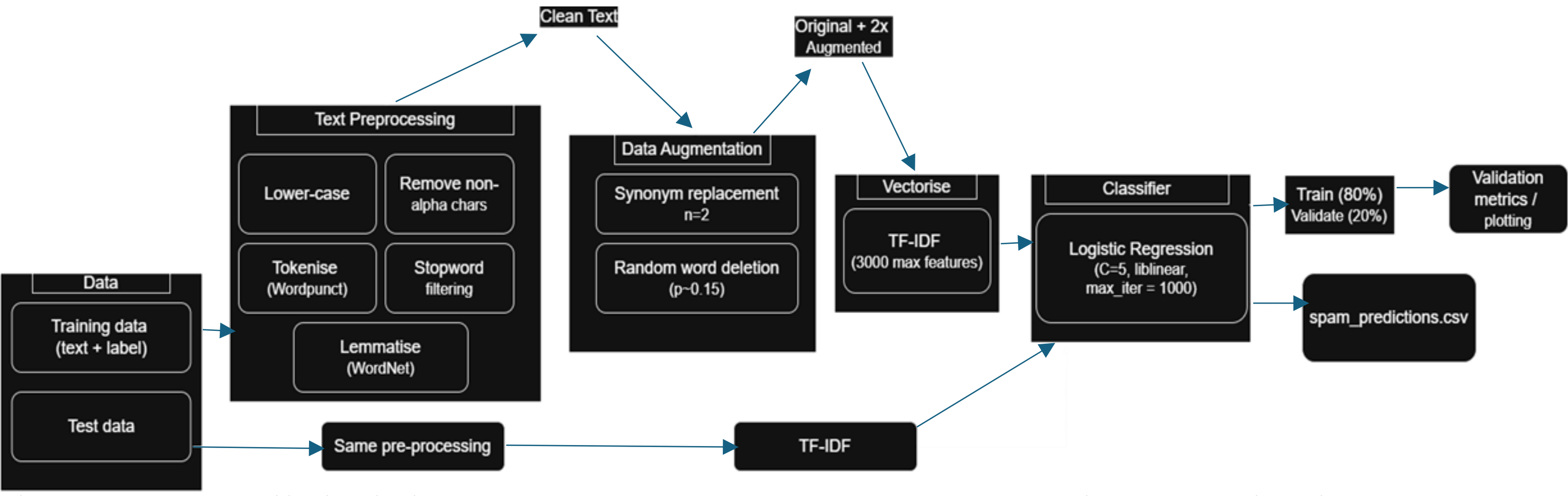


Figure 2 – End to end classification pipeline: Raw messages are cleaned, branched through two augmentation paths, vectorised with a 3 000-term TF-IDF vocabulary, and fed to an L2-regularised Logistic-Regression model (C = 5, liblinear). The augmented corpus is split 80 / 20 for training and validation; the trained model then produces the final spam_predictions.csv.

Results

Evaluation metrics (Lecture 21)

- Confusion-matrix counts (TN, FP, FN, TP)
- Precision = TP / (TP + FP)
- Recall (sensitivity) = TP / (TP + FN)
- F1-score = 2 · (Precision · Recall)/(Precision + Recall)
- Specificity (ham-recall) and derived False-Positive Rate = FP / (FP + TN)
- Overall Accuracy

Table 1: Validation-set performance across models:

Model	Accuracy	Spam Precision	Spam Recall	Spam F1
Keyword rule	0.63	0.36	0.33	0.34
Naïve Bayes + BoW	0.95	0.89	0.94	0.91
LogReg + TF-IDF + Aug.	0.998	0.99	1.00	0.99

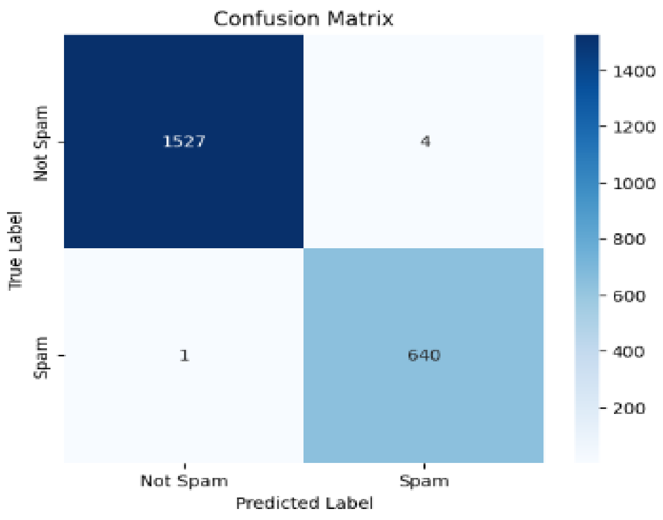


Figure 4 – Confusion matrix for the final spam detector on the validation set. Out of 2 172 messages the model mis-labels only five: four legitimate mails flagged as spam (upper-right) and one spam missed (lower-left), yielding 0.998 overall accuracy.

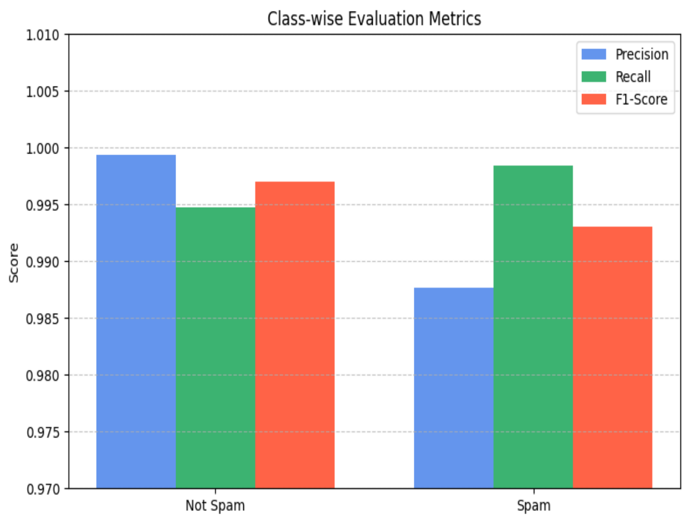


Figure 3 – Per-class precision, recall and F1 for the final spam detector. Both ham and spam achieve near-perfect scores (≥ 0.99).

Residual validation errors

Table 2: 3 mislabels examples are shown below:

Type	Example snippet	True	Predicted
FP	"... *urgent* internal security update ..."	Ham	Spam
FP	"subject president com exciting news ..."	Ham	Spam
FN	"... goodbye gilt chimiquepretend... (obfuscated URL stream)"	Spam	Ham

The False positives are short mails containing words such as "urgent" or "security", but the one false negative was an unusual message lacking typical spam cues.

Test-set submission

The final Logistic-Regression classifier, trained on the entire augmented corpus, produced predictions for spam_detection_test_data.csv, which I saved for submission.

Discussion

The performance of the pipeline spam detection on the validation set can be traced to three key design choices. Firstly, the use of TF-IDF provides good lexical weighting, terms that occur frequently in only a handful of messages, like "unsubscribe" or "voucher", receive big positive weights, where commonly found words contribute little. Replacing Naïve Bayes raw term counts lifted spam precision to 0.99 from 0.89 as seen in table 1, while preserving the recall. Secondly, the linear decision state of L2-regularrisied Logistic Regression proved effective for the task. Once each token is re-weighted by its IDF, the spam and ham classes become much more linearly separable; as seen in figure 3 and examples of provided in table 2. To analyse that, only one example of real spam slipped through, but four examples of false positives, this could be problematic as the model makes more errors misclassifying emails you actually want to receive. Overall, this lead to the conclusion that adding a more complex non-linear model was unnecessary.

Thirdly, targeted data augmentation was able to widen the decision margin, replacing two random words in every message encouraged the model to focus on semantics rather than the surface forms. On the other hand, deleting roughly 15% of words forced robustness to fragmented lines that are common in SMS.

To further analyse the residual error, a pattern emerges. All four false positives were business style e-mails that contained high-risk tokens such as *"urgent"* or *"security"*. In the absence of contextual cues - such as email history, the model and flags them. The false negative was a unclear spam consisting of pseudo-random words and a hidden URL; I deduce that because most tokens were meaningless, its TF–IDF profile resembled legitimate mail. Both cases emphasise that purely lexical evidence, however well weighted, is sometimes not enough.

Assessing the bias and robustness, the system displays no meaningful skew against legitimate traffic: specificity stands at 0.997 and the FP rate at only 0.26% (figure 3). Despite this, the frozen vocabulary after training brings risks as language and spam evolves, retraining or online updates could mitigate this flaw in the model.

When compared with the two baseline iterations, final pipeline strengths are clear. The keyword model achieved 63% accuracy, evidencing the limitations of hard coded heuristics. Improving from there: Naïve Bayes, equipped with bag of words counts, pushed accuracy to 95% (table 1). However, this good performance still produced 38 mis-classified messages. Likely because, every occurrence of a word has the same importance, no matter how rare it is. This is where the TF-IDF differs and shines in this setup, as it integrates the words distinctiveness, and potentially that's the difference in the error count of 5 versus 38.

Several limitations are present in the model. Currently, it ignores unseen words (Lecture 18), which means new slang, misspellings or spam tactics might pass through. Other techniques like character level models could help with this.

Also, there is a lack of attention to word order or sentence structure, as a result it may struggle with more subtle spam. For this, more advanced models like RNNs or transformers that can understand context and flow are useful.

In summary, this study has demonstrated that classic NLP techniques, when tuned systematically, can be effective for classifying spam in moderate sized corpuses. Achieving 0.998 accuracy with only five validation errors depicts the value of pre-processing, discriminative weighting and meaningful data augmentation.

Task 2

Introduction

Face alignment refers to the task of locating pre-defined facial landmarks such as the eyes, nose, mouth corners in images. It's present in many vision tasks. This section describes a machine learning system for robust face alignment.

The model employs a regression-based approach (Lecture W04_L2). Images are pre-processed by resizing to 96x96 pixels and converting to grayscale. Histogram of Oriented Gradients (HOG) features are then computed to capture local shape and appearance. Data augmentation, including horizontal flipping (with landmark adjustments via FLIP_PAIRS) and brightness variation, was applied to the training set. A Ridge Regression model, with its alpha parameter optimized to ~75.0 via cross-validation, was trained on these HOG features to predict the five (x,y) landmark coordinates.

The final model achieved a Mean Point Error (MPE) of 4.36 pixels on the validation set (evaluated on the original 256x256 scale, after 96x96 processing), showing HOG features, regularised linear regression, and data augmentations effectiveness for this task.

Methods

Overview

The goal was to accurately localise five facial landmarks. Utilising Histogram of Oriented Gradients (HOG) features from pre-processed images to train a Ridge Regression model, with data augmentation enhancing robustness.

The face_alignment_training_images.npz dataset (images and landmark coordinates) was used for development in an 80% training 20% testing split, and face_alignment_test_images.npz (images only) for final predictions.

Image Preprocessing / Data Augmentation

A pre-process function prepared images for feature extraction in the following steps:

1. Grayscale Conversion: RGB images were converted to grayscale (cv2.cvtColor).
2. Brightness Augmentation (Training Only): Applied with 50% probability, randomly adjusting contrast (alpha 0.7-1.3) and brightness (beta -20 to 20) using cv2.convertScaleAbs.
3. Image Resizing: All images were resized to 96x96 pixels (IMG_H, IMG_W) using cv2.resize – found 96x96 yielded best results on validation.
4. Landmark Scaling (Training Only): Training landmark coordinates were scaled proportionally to the 96x96 dimensions.
5. Horizontal Flipping Augmentation (Training Only): Applied with 50% probability. Images were flipped (cv2.flip), x-coordinates adjusted ((IMG_W - 1) - x), and symmetric landmark identities swapped using FLIP_PAIRS = {0: 1, 1: 0, 3: 4, 4: 3} ((Lecture W04_L2).
6. Pixel Normalization: Pixel values were scaled to [0,1].

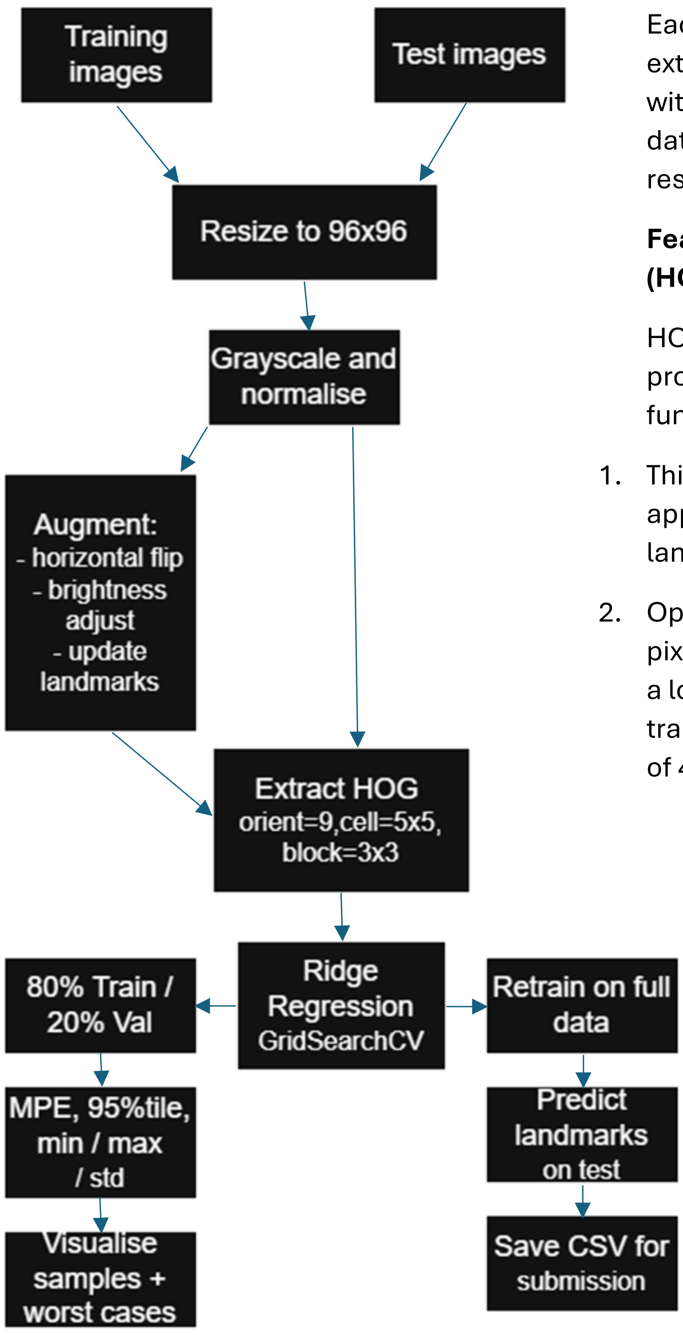


Figure 5 - Face-alignment pipeline. Images are resized, normalised and optionally augmented (horizontal flip with landmark swap and brightness jitter) before HOG descriptors are extracted. A Ridge-regression model (α selected via GridSearchCV) is trained on an 80 / 20 train-validation split; metrics and visual checks are produced, then the model is retrained on all data and used to output landmark predictions for submission.

Each training image yielded two versions for feature extraction: one with basic preprocessing and another with probabilistic augmentations, doubling the training data. Test images underwent only grayscale conversion, resizing, and normalization (test_preproc function).

Feature Extraction: Histogram of Oriented Gradients (HOG)

HOG features were extracted from the 96x96 pre-processed images using skimage.feature.hog (hogify function) (Lecture W04_L1).

1. This was chosen, as HOG captures local shape and appearance robustly, so I deemed it suitable for landmark localization.
2. Optimal parameters after tuning: orientations=9, pixels_per_cell=(5,5), cells_per_block=(3,3) – tuned these a lot to find optimal balance, block_norm='L2-Hys', transform_sqrt=True. These parameters yielded an MPE of 4.36 pixels (on 256x256 scale).

Prediction Model: Ridge Regression

Landmark (x,y) coordinates were predicted using Ridge Regression (sklearn.linear_model.Ridge).

1. Ridge Regression (L2 regularization) prevents overfitting with high-dimensional HOG features and is computationally efficient.
2. Hyperparameter Tuning: GridSearchCV with 5-fold cross-validation and 'neg_mean_squared_error' scoring selected the optimal alpha from [1, 25, 50, 75, 100, 125, 150]. The best alpha found was 75.0.
3. Training: The model was initially trained on an 80% split of the augmented, HOG-featured dataset. For final predictions, it was retrained on the entire augmented training set using the best alpha. Landmark coordinates were flattened for regression.

Test Set Prediction and Output Scaling

Test image HOG features were fed to the retrained model. Predicted 96x96 scale coordinates were scaled up to the original 256x256 image dimensions (scale = ORIG_H / IMG_H) and saved.

Results

Experimental Results

To evaluate the face alignment, comparing different feature extraction methods and the impact of data augmentation was observed. The main metric analysed was Mean Point Error (MPE), calculated as the average distance in pixels from predicted and ground truth landmark coordinate in the validation set, scaled to original resolution.

Validation Set Performance

The MPE of different approaches on the held-out validation set is summarised in Table 1. The SIFT model served as a baseline (Lecture W04_L1), followed by a model using HOG, finally settling on the refined model incorporating HOG with data augmentation.

Table 1

Model/Feature Set	**Mean Point Error (MPE) on Validation Set (pixels)**
SIFT + Ridge Regression	8.77
HOG + Ridge Regression	5.12
HOG + Data Augmentation + Ridge Reg.	4.36

Using HOG descriptors with horizontal-flip and brightness augmentation (Lecture W04_L2), the Ridge–regression model (optimal α = 75.0 from cross-validation) produced the strongest results on the held-out validation set. The mean point-wise error across all landmarks was 4.36 px, with a median of roughly 3.23 px, indicating a tight error distribution around the mean. The best individual prediction missed its ground-truth location by only 0.03 px, whereas the worst miss was 31.86 px; nonetheless, 95 % of all landmark predictions lay within 10.49 px of the correct position.

Qualitative Results

Visual examples of the final model's predictions on sample validation images are presented in Figure 6, with green markers indicating ground truth landmarks and red markers indicating the predictions.

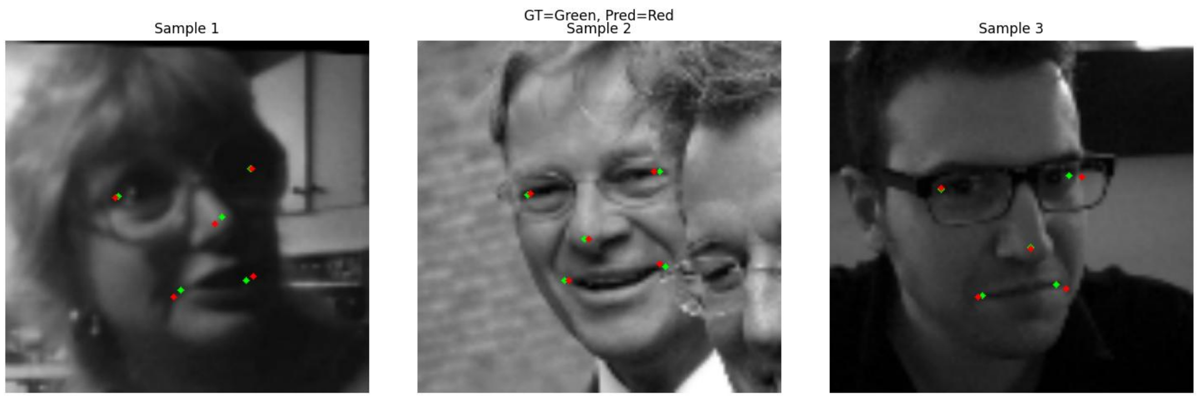


Figure 6: validation examples for face alignment. Three randomly selected images with ground-truth landmarks (green) and predictions (red) illustrate the close correspondence achieved by the HOG + augmentation model.

The distribution of errors provides further insight into the model's performance. Figure 7 displays a histogram of the Mean Point Error per image on the validation set and a boxplot of the pointwise Euclidean distances (Lecture W04_L2) for all landmarks from the final model

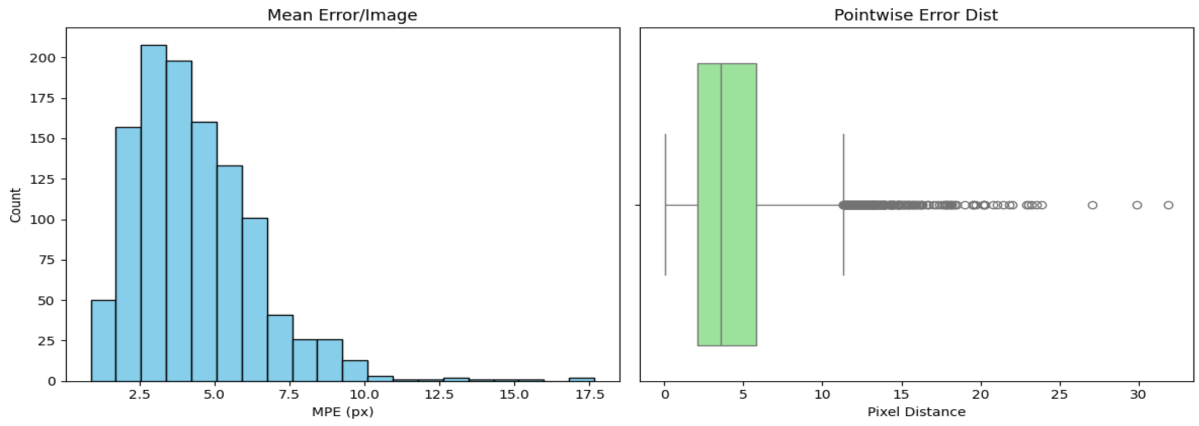


Figure 7: Validation error distributions: Histogram (left) shows most faces have 2–6 px mean error; boxplot (right) places the median point error near 3 px with few outliers beyond 10 px.

The histogram (Figure 7, Left) indicates that the majority of images have a low MPE, with the distribution peaking around 2.5-5 pixels. The boxplot (Figure 7, Right) shows that the median

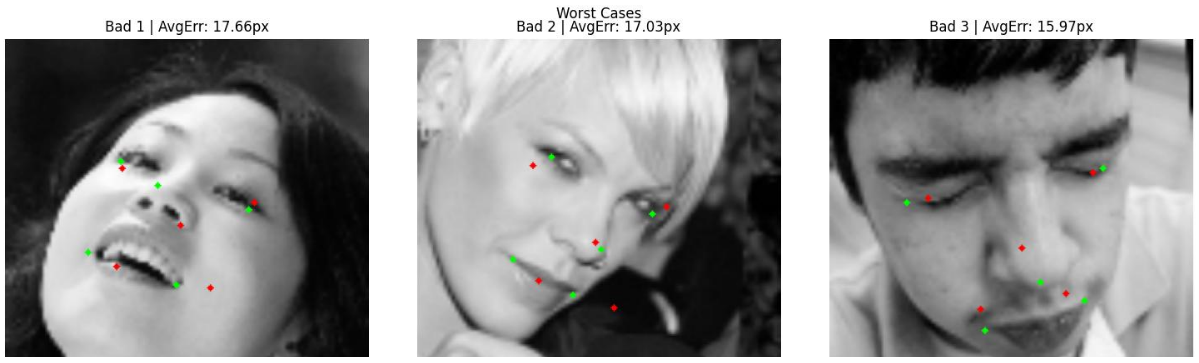


Figure 8: Largest-error examples. Three outlier faces (mean error ≈ 16–18 px) illustrate failure modes under extreme pose, expression or occlusion; ground-truth points are green, predictions red.

pointwise error is approximately 3.23 pixels, with most errors concentrated below 7.5 pixels, although some outliers exhibit larger deviations.

Model limitations are visualised in Figure 8, displaying three of the worst-performing predictions, showing discrepancies between predicted (red) and ground truth (green) landmarks. The final HOG + Data Augmentation + Ridge Regression model, retrained on the entire augmented training dataset using the optimal alpha of 75.0, was used to predict on the face_alignment_test_images.npz dataset. These predictions, scaled (256x256), were saved.

Discussion

The developed system, utilising HOG features, Ridge Regression and data augmentation achieved the lowest Mean Point Error of 4.36 pixels on the 256x256 validation set. These results show the pipelines effectiveness in localising facial landmarks as seen in figure 6. There was iterative and comparative design to aid model choice and development. Initial analysis of SIFT versus HOG identified which route could offer the most results on the surface. With the SIFT yielding 8.77MPE and HOG coming in at 5.12MPE (table 1), showing HOGs superior ability to capture local information for this task. Adding data augmentation, flipping images (and correctly adjusting the landmark labels using FLIP_PAIRS) and tweaking the brightness, the most notable improvement was developed as it helped with different poses and lighting conditions while not being too computationally expensive to run. Also, resizing the images to 96x96 gave enough detail for good performance without becoming too slow to run. This enabled the Ridge Regression model (α=75.0 via cross-validation) to learn more generalisable representations less sensitive to pose and lighting variations.

Despite a low MPE (4.36px), the error distribution (figure 7) shows a 95th percentile of 10.49px and a max of 31.86px. Analysing high-error cases (Figure 8, average errors ~16-17px), this indicates that the model mainly struggled with images that had diverse head poses/rotations, strong or unusual facial expressions, partial occlusions, and challenging lighting conditions. These likely differ from dominant HOG-learned patterns, even with the augmented training set. These instances likely deviate significantly from the dominant patterns learnable by HOG features from the augmented training data. The handcrafted nature of HOG may limit its ability to capture highly abstract relationships necessary for these variant cases. Also, a potential bias towards more common frontal, well-lit faces might exist, leading to performance degradation on underrepresented inputs. Moreover, since HOG is a handcrafted feature extractor – it might not be flexible enough to capture abstract facial patterns. Equally, it may bias towards common face types in training, which could explain why it struggles with unusual ones.

In summary, the final system was effective at performing face alignment. But, future enhancements could improve its limitations, such as including more augmentation, like adding rotations. HOG was effective, but exploring pre-trained Convolutional Neural Networks (CNNs) could capture more complex visual patterns, potentially improving accuracy on challenging cases, granted at a higher computational cost. Exploring non-linear methods might also yield performance benefits. Finally, using cascaded regression to refine landmark estimates, could enhance precision, especially for landmarks that are inherently harder to localize in a single step (Lecture W06_L1).