Machine Learning - Final Project: Country-211

Itai Mizlish(211821913) Eli Kozinetz(314666561)

Introduction

In this project, we explore a classification task using a subset of the Country-211 dataset, which consists of images labeled by their country of origin. To enable focused experimentation and analysis, we restrict our study to five selected countries: Israel, the United States, Denmark, Japan, and Nigeria. The goal is to evaluate and compare the performance of various pre-processing and modeling techniques for image-based country classification.
We experiment with several input representations, including **raw RGB pixel vectors**, **grayscale-converted images**, and **high-level embeddings extracted from a pre-trained VGG network**. To better understand the structure and separability of the data set, we applied **Principal Component Analysis (PCA)** to reduce dimensionality and visualize the results. We also used **Cosine-Similariry** on the vectorized types of data in order to find out which labels are similar to one another and which are not.
For the classification task, we benchmark a variety of models: classical machine learning algorithms such as **Support Vector Machines (SVM), Random Forests, and AdaBoost; custom-built multilayer perceptions (MLPs); and a fine-tuned ResNet-50 deep learning model**. We evaluated the effectiveness of each approach across the different input representations, aiming to identify the most accurate and efficient classification pipeline. Through this work, we also aim to gain deeper insight into the trade-offs between traditional and deep learning approaches, and highlight the challenges inherent in working with image-based classification problems.

Exploratory data analysis

We investigate a five-class subset (**COUNTRY-5**) of the Country211 dataset to assess data balance, image characteristics and pixel statistics prior to model development. The subset comprises: Israel (**IL**), United States (**US**), Germany (**DE**), Nigeria (**NG**), and Japan (**JP**).

Dataset and Split

The official train/validation/test splits were filtered to retain only images whose integer labels map to the selected ISO codes. Table 1 confirms an exactly balanced design (150/50/100 per class).

Table 1: Image count per split after filtering (COUNTRY-5).

Country	**Train**	**Val**	**Test**
DE	150	50	100
IL	150	50	100
NG	150	50	100
US	150	50	100
JP	150	50	100
Total	750	250	500

Figure 1 visualises the same counts.

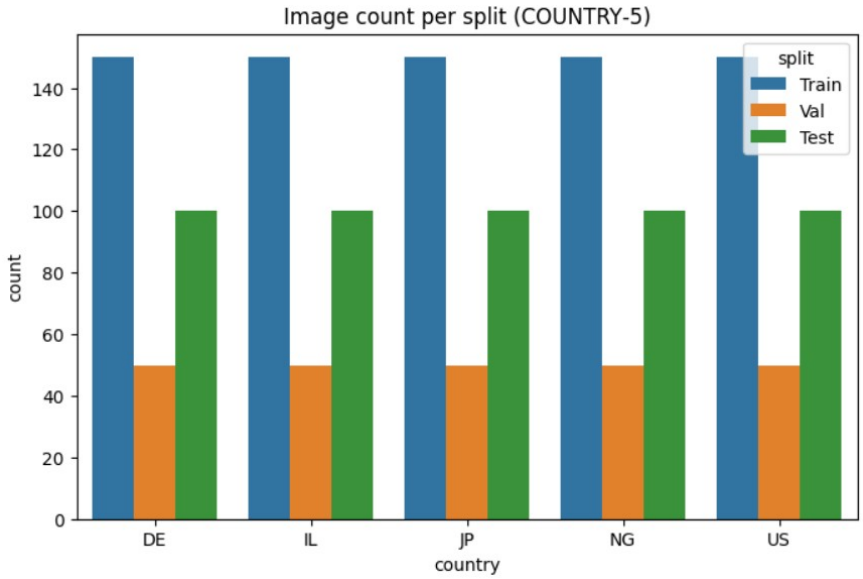


Figure 1: Balanced image distribution across splits (COUNTRY-5).

Qualitative Inspection

Representative samples reveal substantial intra-class diversity (Figure 2). Images span natural scenes, objects and portraits, underscoring the need for strong visual invariance in downstream models.

(a) DE (b) IL (c) NG (d) US (e) JP

Figure 2: Three random training images per class.

Image Resolution and Aspect Ratio

Descriptive statistics for the 600 training images are summarised in Table 2; histograms appear in Figure 3. Over 70 % of images are exactly 500 px wide, yet heights vary between 300 px and 500 px, yielding aspect ratios from 0.56 to 1.98. A fixed-resolution pre-processing pipeline (e.g. center crop to 224×224) is therefore recommended.

Table 2: Resolution statistics (train split).

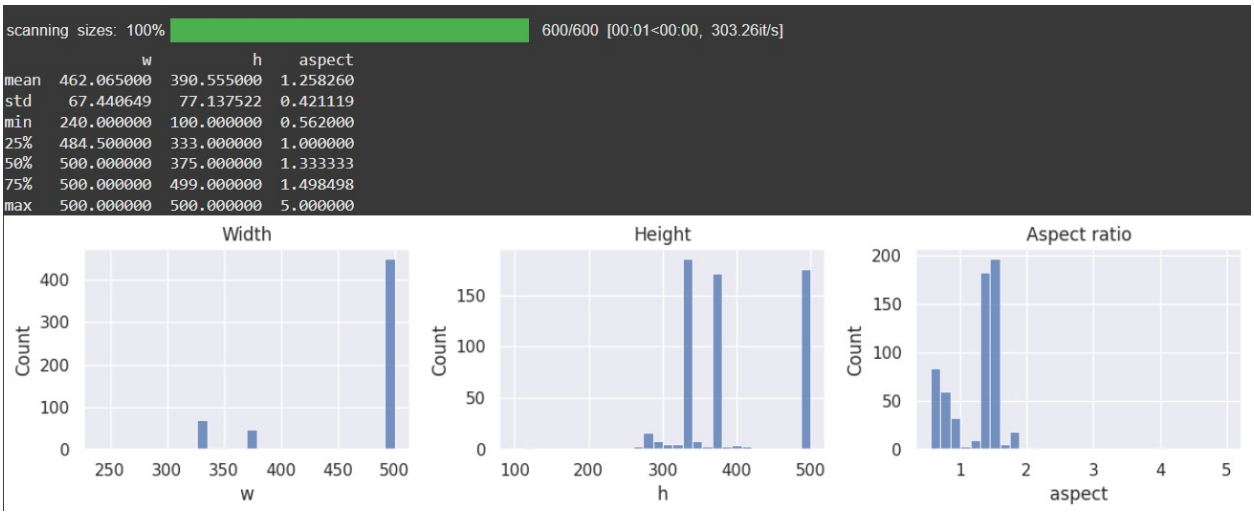


Figure 3: Histograms of widths, heights and aspect ratios.

Pixel Intensity Distribution

Mean and standard deviation per RGB channel (based on 2 k random images) are

$$\boldsymbol{\mu} = (0.4665,\ 0.4411,\ 0.4051), \qquad \boldsymbol{\sigma} = (0.2422,\ 0.2349,\ 0.2387).$$

These values will be used to standardise inputs during training.

Pre-Processing

In order to use common exploration, analysis and training logic from images, which are under the hood a 3 channel matrix, where each channel is represented by a color channel, Red, Green, Blue, respectively, we had to pre-process the given images from the dataset into a form of data we can work with.
We used 4 techniques for that:

- **Fixed Image Size**: In order for the analysis and models to work properly, we make sure the number of features across the datasets is the same.
- **Flattening**: By taking the matrix and flattening it into a single vector, we get a set of features. However, a major disadvantage of it is that the relationship between pixels is lost.
- **Convert To Grayscale**: By taking an RGB image and converting it to a gray image, the channel count is reduced from 3 to 1, reducing memory and simplifying flattening, at the cost of color data.
- **Vector Embeddings**: Given an image, we can use the deep learning model **VGG** to produce embedding vectors, where each vector represents some value of a concept.

The preprocessing pipeline consists of applying these techniques in order to produce 3 distinct datasets, where each data-frame is a form of data used to analyze and train on. These will be considered the **vectorized datasets** of the project.

- **Flattened RGB**: After fixing the image size and applying flattening, we get a dataset in which each sample is a vector. Each value of the vector represents a value from a channel where it once was. A big disadvantage is the loss of the relationship between columns. Each sample becomes a vector of length 49152.
- **Flattened Gray**: Same as Flattened RGB, but each image is also converted to Grayscale before flattening it. Each sample becomes a vector of length 16384.
- **VGG Embeddings**: Each image gets a vector embedding using **VGG**. Each sample becomes a vector of length 25088.

There is also a 4th dataset: the original image dataset itself, which is not vectorized at all. It will be reserved for training the **ResNet-50** deep learning CNN model. Its preprocessing will consist of fixing the image size to 224X224, and normalizing the RGB channel values, since this is the input dimentions ResNet-50 expects.

Discussion

- The dataset is **balanced** across the chosen classes, avoiding the need for re-sampling at training time.
- Resolution variability is modest but non-negligible: a unified resize/crop step is advisable.
- The per-channel statistics align with ImageNet norms; transfer learning with pre-trained CNNs is feasible.

Conclusion

The COUNTRY-5 subset provides a clean, balanced starting point for multi-class classification experiments. The EDA confirms both the strengths (balance, variety) and the risks (resolution variability) that must guide subsequent model-selection and augmentation strategies.

Question 1: Assessing Similarity – How Distinct Are the Representations of Each Country?

In order to answer the question of similarity, we used two techniques: **PCA** plotting and **cosine similarity**.

PCA

We Used PCA in order to visualize the closeness of labels in the feature space. Since the vectored data has more then 3 dimensions, it is important to apply PCA to visualize any closeness and similarity of labels.

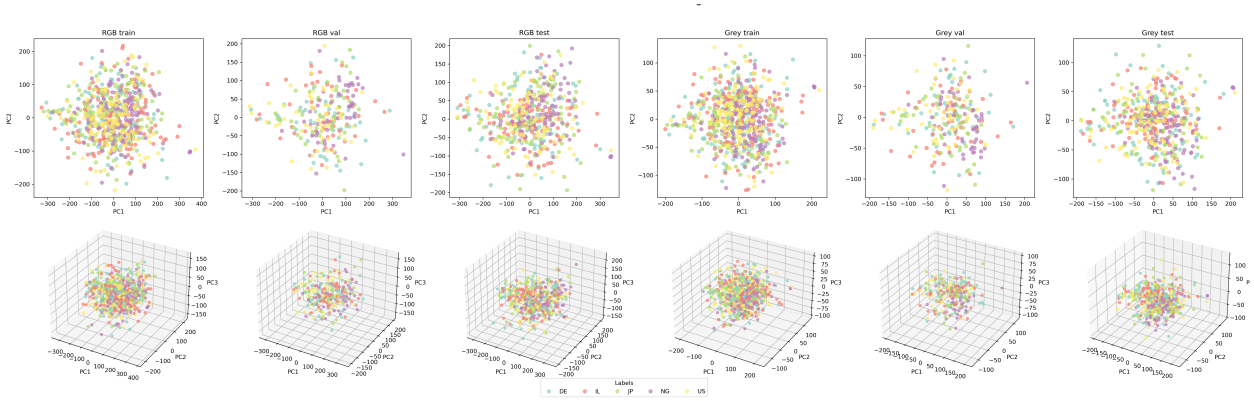


Figure 4: PCA results of all vectorized data, not including VGG in this figure.

The results across the vectorized data show a lot of noisy placement of the values, meaning that its very hard to discern any separability of labels based on the pixel values alone.

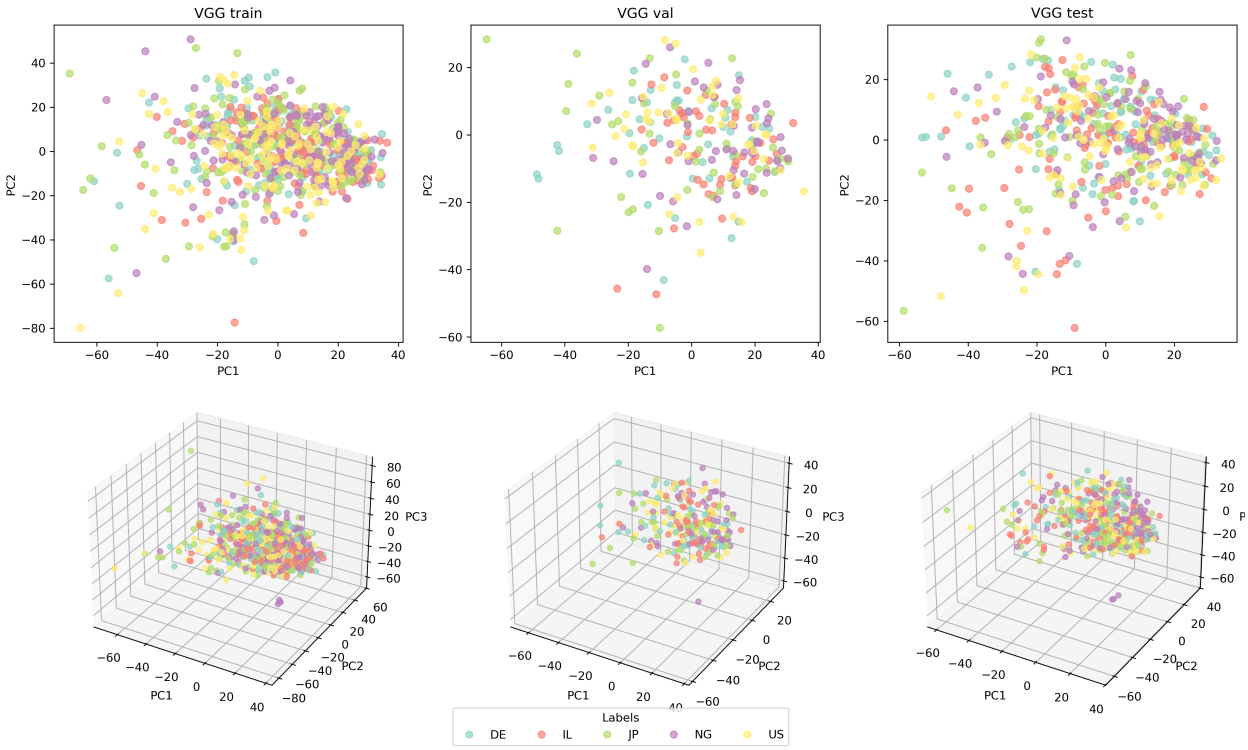


Figure 5: PCA results of VGA data.

The results on the VGG embeddings show a different result, where most values are centered around in the same place, with a few outliers of each label.

Overall, the results of PCA similarity tell us that its very difficult to see any clustering across vectorized data, which VGG improving the results slightly. This suggests that

Cosine Similarity

We ran the cosine similarity algorithm on all 3 vectorized datasets in order to check if it can identify Country cross similarity. Meaning which country is most similar to which, and difference as well.

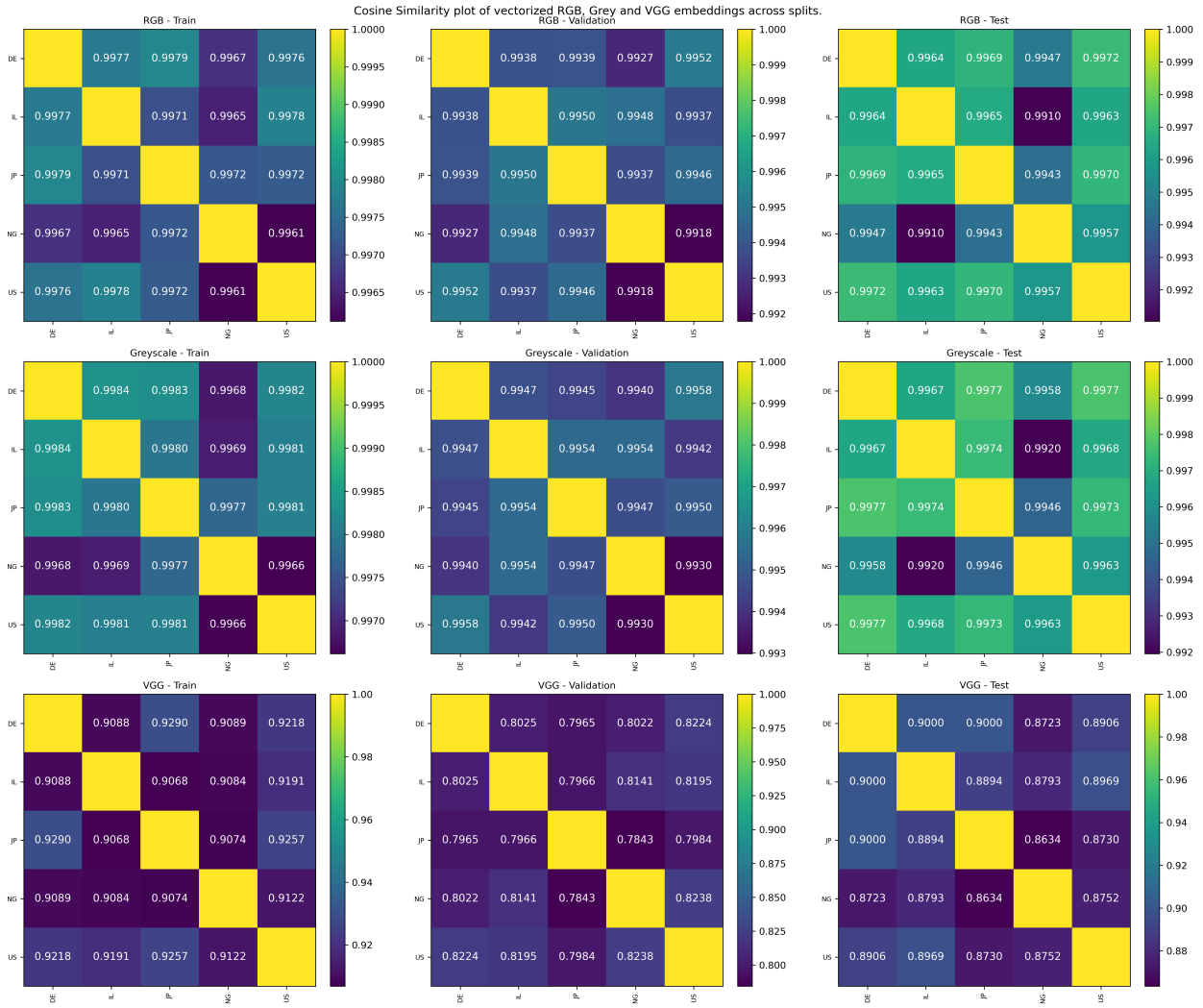


Figure 6: Cosine Similarity.

As we can see, cosine similarity across the flattened vectors is not relevant since it just means similarity across individual pixels. In order to solve this, we've done the same thing but with the VGG embeddings, which semantically makes more sense, and we got more meaningful results. For example, we see that across splits, the results are more or less in the same distribution. In addition to that, we notice for example that western countries are more similar to each other, but Nigeria and Japan are the most different across countries.

Conclusions

- **PCA on raw RGB/grey vectors** yields heavily overlapping point-clouds; pixel-level variance dominates, masking any country-level clusters.
- **VGG embeddings** compress semantically meaningful information, reducing noise and producing a tighter central cluster with a handful of country-specific outliers.

- **Cosine similarity of flattened pixels** is uninformative: scores are almost uniformly high (~ 0.995), reflecting low-level colour similarity rather than geographic cues.
- **Cosine similarity of VGG embeddings** exposes structure: Western countries (DE, FR, US) cluster together, while JP is moderately distant and NG forms the most distinct group.
- Across train/val/test splits, the relative similarity pattern is stable, indicating that the observed relationships are robust and not an artefact of a particular split.

Overall insight. Low-level pixel statistics alone are insufficient to distinguish the five countries, but pretrained CNN features (VGG) provide a semantically enriched representation that reveals subtle, yet consistent, inter-country relationships. This suggests that downstream models should operate on high-level embeddings rather than raw pixel vectors when geographic origin is the prediction target.

Challenges

- *Curse of dimensionality.* Flattening $128 \times 128 \times 3$ images produced 49152-dimensional vectors, making distance metrics noisy. **Solution:** Applied PCA (3 components) to visualise and confirm the absence of structure before switching to lower-dimensional CNN embeddings.
- *Semantic gap.* Pixel-space similarity reflects lighting and colour rather than content. **Solution:** Generated 25088-D VGG penultimate-layer embeddings that encode shape and texture, bridging the gap.
- *Metric saturation.* Cosine similarity on raw vectors clustered near 1.0, hiding differences. **Solution:** Re-computed similarity on VGG embeddings, which produced a wider dynamic range and meaningful separation.
- *Visual clutter in PCA plots.* Overplotting obscured clusters. **Solution:** Used transparency (`alpha=0.6`) and jitter, and provided both 2-D and interactive 3-D views for clarity.

Question 2: Exploratory Analysis of RGB Histogram Features – Can Global Colour Distributions Predict Image Origin?

Research question. Can low–level colour statistics alone discriminate the geographic origin of an image among the five selected countries (DE, IL, JP, NG, US)? A 24-dimensional RGB histogram (8 equal-width bins per channel) is evaluated as a classic global descriptor.

```python
import numpy as np, pandas as pd

def mean_hist_df(ds):
    X = np.stack(ds["hist"])                    # (N,24)
    countries = [label2country(int(l)) for l in ds["label"]]
    df = pd.DataFrame(X, columns=[f"bin_{i}" for i in range(24)])
    df["country"] = countries
    return df.groupby("country").mean()

mean_df = mean_hist_df(ds_train_h)
mean_df.head()
```

country	bin_0	bin_1	bin_2	bin_3	bin_4	bin_5	bin_6	bin_7	bin_8	bin_9
DE	32043.746667	23070.273333	24036.473333	24324.880000	22353.806667	19264.680000	16070.593333	17287.440000	31417.333333	25174.960000
IL	26478.620000	21223.793333	21132.693333	22468.333333	23264.773333	23746.780000	17314.313333	18905.766667	30689.346667	23424.680000
JP	38409.053333	26489.280000	22214.553333	22628.846667	22238.073333	18857.173333	13851.200000	14171.913333	40865.286667	28181.646667
NG	15536.420000	22276.093333	25109.786667	28317.100000	25695.460000	24298.100000	16572.433333	14700.246667	15371.580000	25264.020000
US	25908.213333	25346.620000	26528.973333	25162.100000	22678.793333	23671.880000	15895.680000	16826.893333	28669.966667	30012.220000

5 rows × 24 columns

Figure 7: Mean 24-bin RGB histogram for each country (train split).

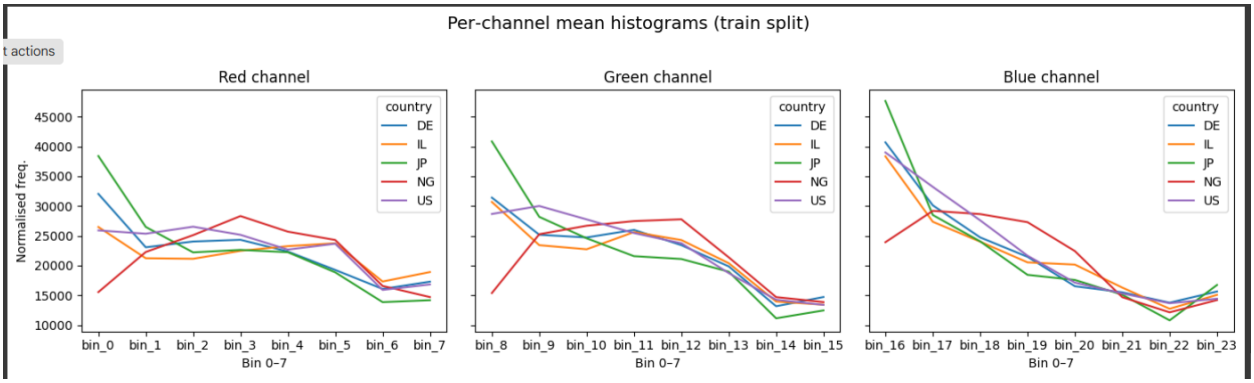


Figure 8: Per-channel mean histograms. JP shows the highest peak in the blue bins, whereas NG exhibits elevated red counts.

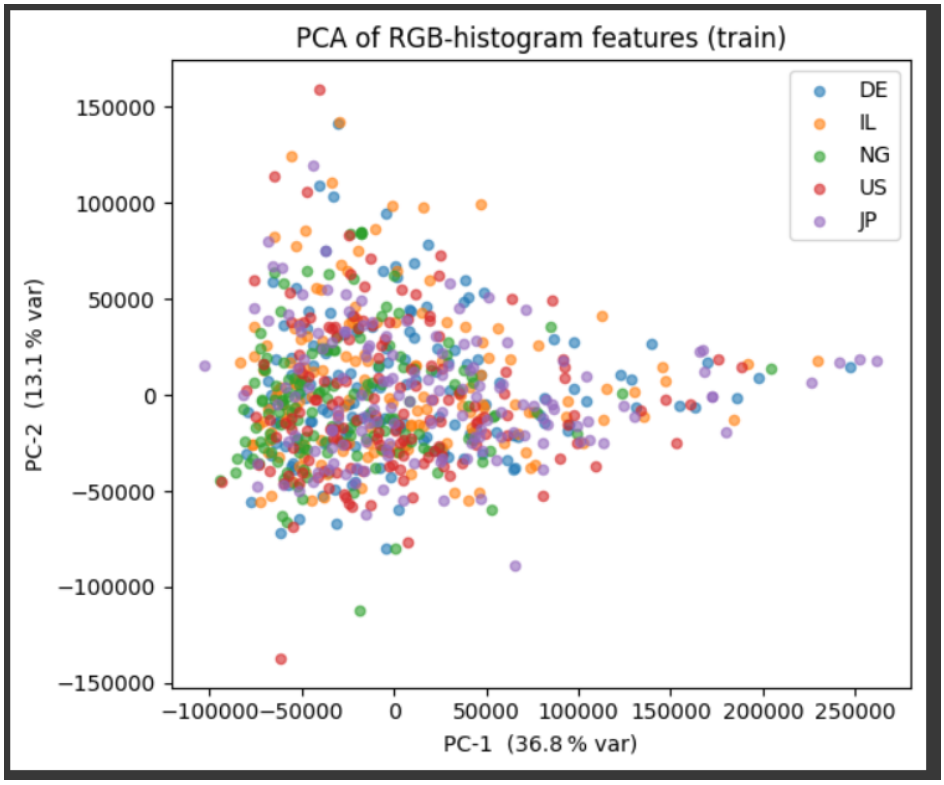


Figure 9: Two–dimensional PCA projection of the 24-D histogram features. The first two components explain about fifty per cent of the variance but class overlap remains substantial.

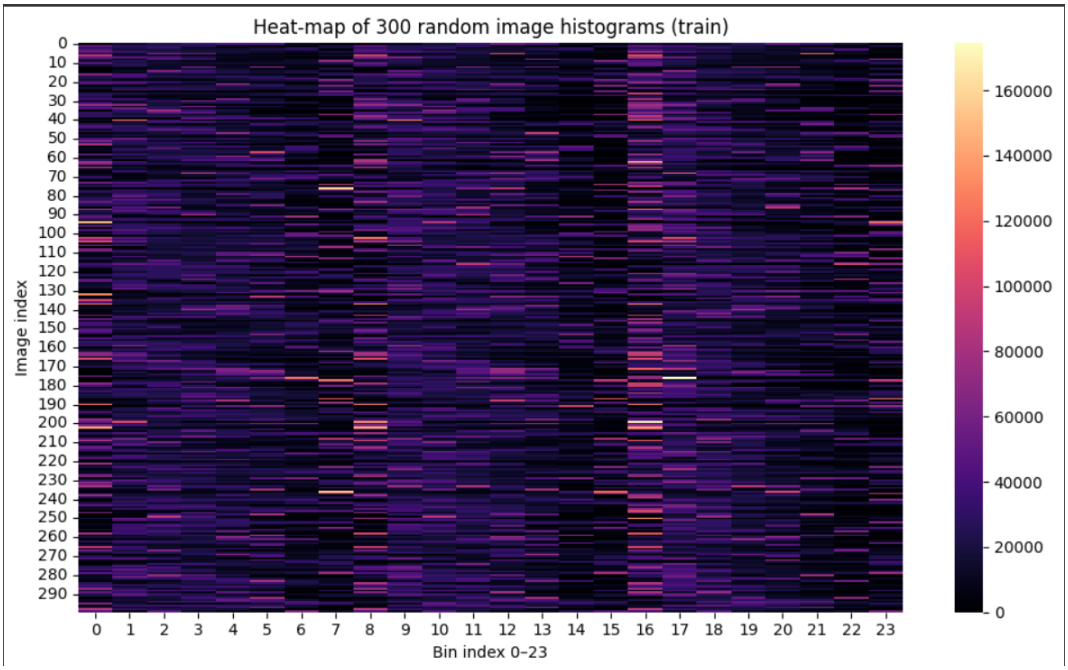


Figure 10: Heat-map of RGB histograms for 300 random training images. Vertical stripes reveal images dominated by single colours.

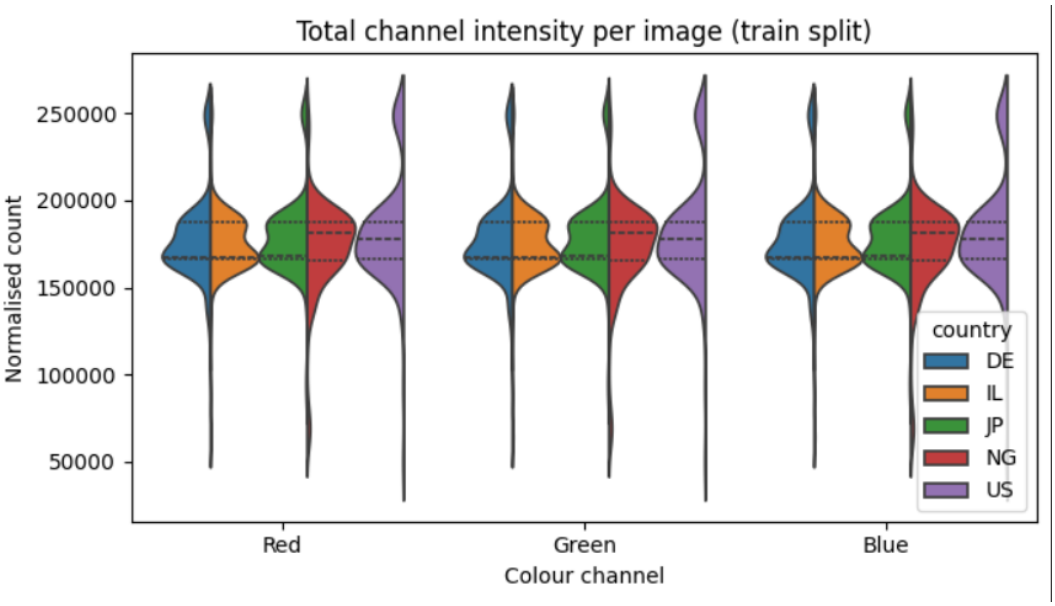


Figure 11: Violin plot of total channel intensities per image (train split).

Baseline classifier. A grid-tuned k-nearest-neighbour (KNN) model with $k = 13$ was trained on the histogram vectors. Table 3 summarises accuracy; Figure 12 shows the confusion matrix.

Table 3: Accuracy of KNN on RGB histograms.

	Train	Validation	Test
Accuracy	0.495	0.240	0.234
Chance		0.200	

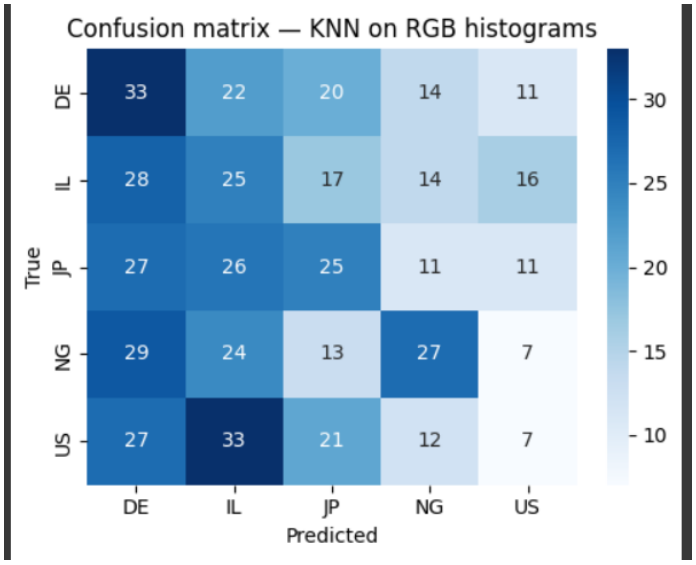


Figure 12: Confusion matrix for the five-way test split using KNN on RGB-histogram features.

Conclusions:

- **Global color distributions contain weak geo-signal.** While RGB histograms slightly outperform chance.
- **Ignoring spatial context limits performance.** As known in image retrieval literature, global histograms ignore spatial pixel arrangement—two images with similar color profiles but different content can be indistinguishable by this descriptor.
- **Sensitive to lighting and quantization noise.** Color distributions vary significantly with illumination and sensor differences, undermining histogram stability.
- **Dimensionality–reduction provides limited benefit.** PCA helped reduce noise and simplify visualization, but did not uncover strong class separability (Figure 9), confirming limited information in colour alone.
- **Spatial and semantic features are necessary.** Findings strongly motivate the use of richer representations—such as CNN embeddings or spatial–color hybrids—that encode texture, shape, and location-based cues, known to improve geo-classification accuracy.

Despite a small improvement over chance, the test accuracy of 23.4 percent indicates that global colour statistics alone are not sufficient for reliable geo-classification. The result motivates the use of richer spatial features (e.g. CNN embeddings) in subsequent experiments.

Challenges:

- Global colour histograms are highly sensitive to **illumination variability and lighting conditions**, leading to inconsistency across images from different environments.

Exposure changes or different sensor calibrations significantly distort the feature distribution.

- Being a **non-spatial descriptor**, the RGB histogram ignores texture, shape, and pixel arrangement—images with different content but similar global colours can appear identical to the classifier.
- The choice of **bin quantization** (e.g., 8 bins per channel) influences descriptor quality: too coarse loses discriminative detail; too fine yields high-dimensional, sparse vectors prone to overfitting or noise sensitivity.
- Using a **high-dimensional histogram** (24-D) increases computational cost and reduces classifier efficiency. Without dimensionality reduction, models like KNN suffer from the "curse of dimensionality."
- Lighting intensity shifts and **quantization noise** across devices and environments introduce additional variation, making global-colour-only features unreliable.

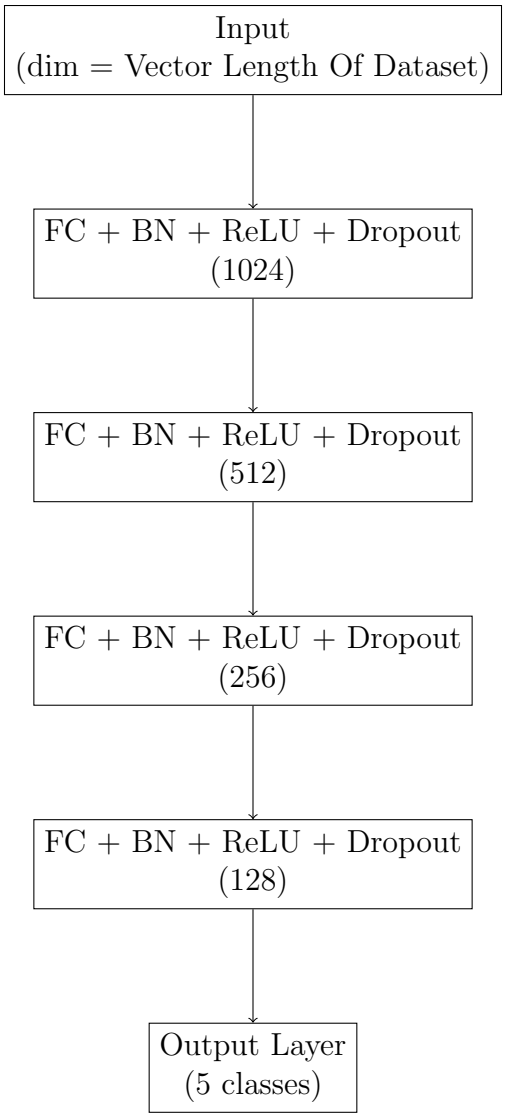
How We Addressed These Challenges:

- We carefully tuned the histogram bin count (8 bins/channel), balancing feature resolution with vector stability to avoid sparsity and overfit.
- Standardised image preprocessing—rescaling and colour normalization—helped reduce illumination variability and maintain consistent histograms.
- Applied **PCA dimensionality reduction** to project 24-D histograms into 2D space (Figure 9), reducing noise and computational complexity while preserving useful variance.
- Adopted a grid-search strategy to optimise the KNN classifier (choosing $k = 13$), helping it generalise better despite the noisy input feature.
- Conducted a **qualitative heatmap analysis** (Figure 10) to visually confirm variations in colour-dominant images and guide interpretation of model failures.

Question 3: Model Evaluation

In the previous sections, we explored several forms of pre-processing image data. The first form of data is representing the images as vectors, with 3 special ways to represent the vectors. The second form of data is the images themselves, with the idea of fine-tuning an existing CNN model in order to compare it against classical ML and MLP models.
A table with all the results will be presented at the end of the chapter.
For the first form of data, the models used are:

- **SVM**: The kernel will be 'RBF', C=1 with 'gamma' scale.
- **Random Forest**: With 100 decision trees voting on the label.
- **AdaBoost**: With 100 estimators.
- **Custom-Created MLP**: Consists of four hidden layers with 1024, 512, 256, and 128 units respectively. Each layer is followed by Batch Normalization, a ReLU activation, and Dropout with a rate of 0.3. The output layer is a linear layer mapping to the number of classes. The model was trained using the Adam optimizer and a learning rate of 10^{-3}.



For the second form of data, we fine-tune the already existing **ResNet-50**, a deep learning CNN model. In the end of the section, a table of all the results is shown.

Vectorized Data - ML and MLP

It is well known that classical ML and MLP models work by receiving vectors where each value is a feature's value.
For each vectorized data type, we trained 4 models.

Flattened RGB

Across RGB data, the model that performed best only based on test accuracy is the Test accuracy with SVM at 26% accuracy. All models have overfitted, with AdaBoost and MLP being the worst models in terms of test accuracy and training time. Oveall results suggest this is an inefficient data to train models with.

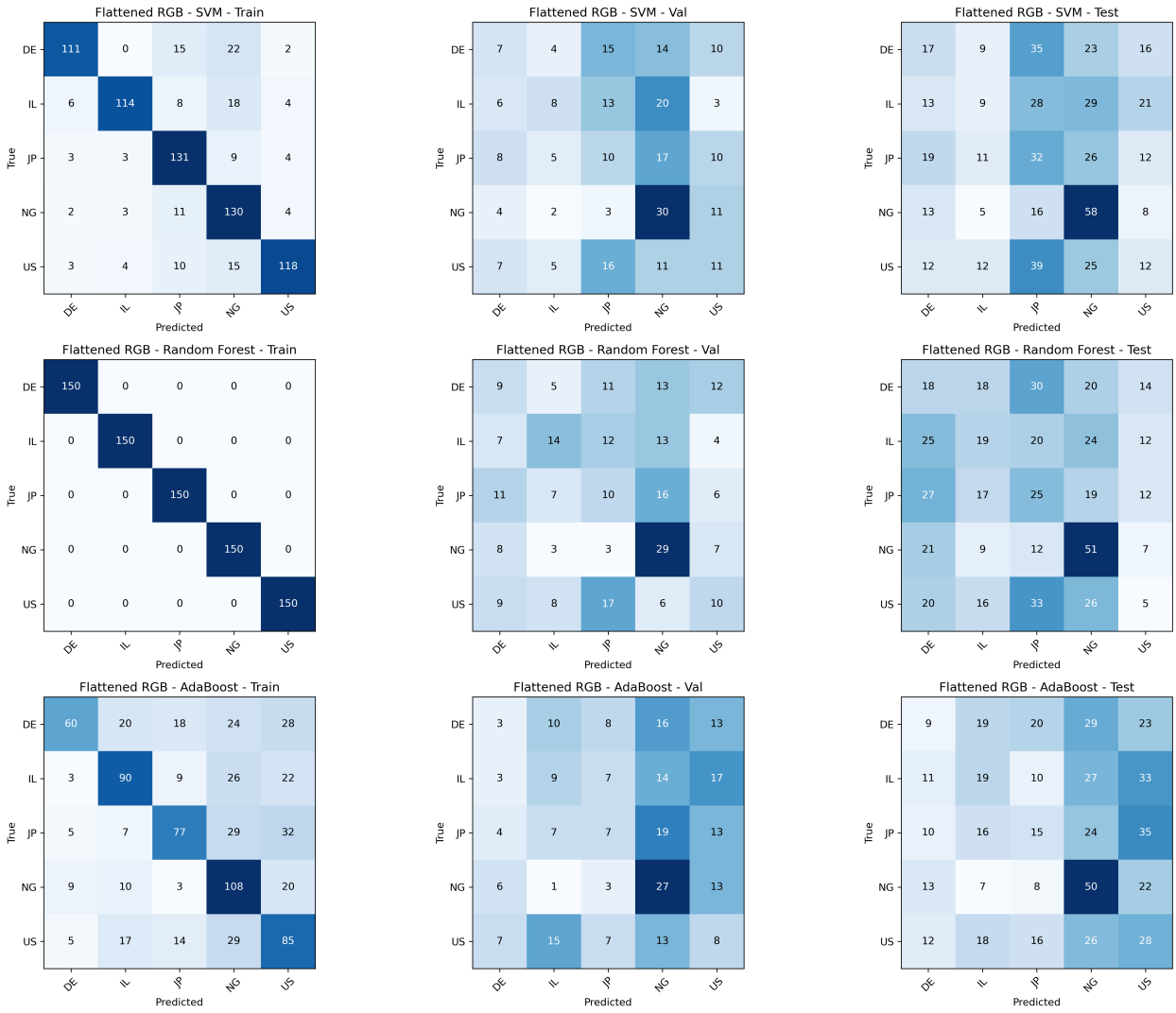


Figure 1: Flattened RGB confusion matrices on ML models.

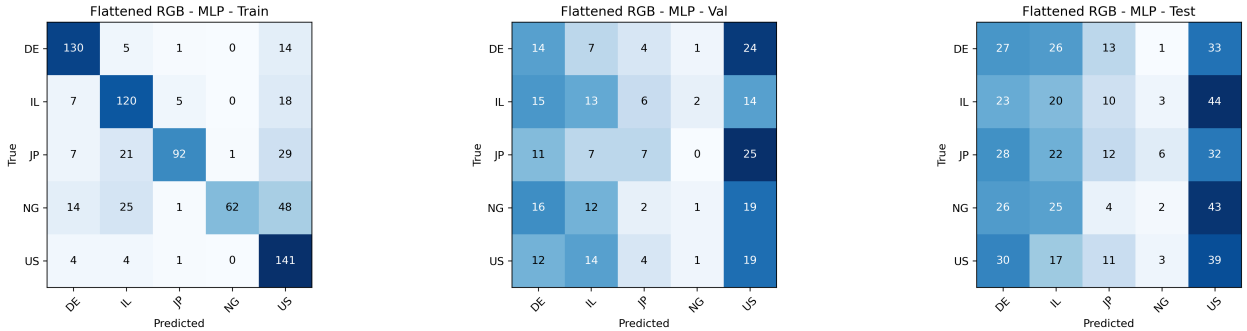


Figure 2: Flattened RGB confusion matrices on MLP model.

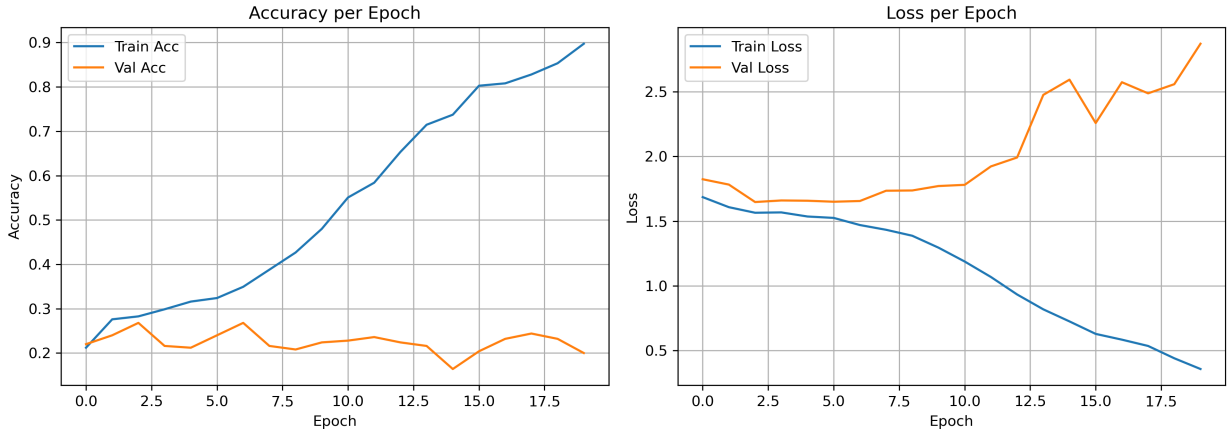


Figure 3: Flattened RGB MLP model training curves.

Flattened Grayscale

Model performance with this data type has worsened across all models, with only training time improving since we have less features to train on. In addition, it looks that the MLP model is biased towards the DE label.

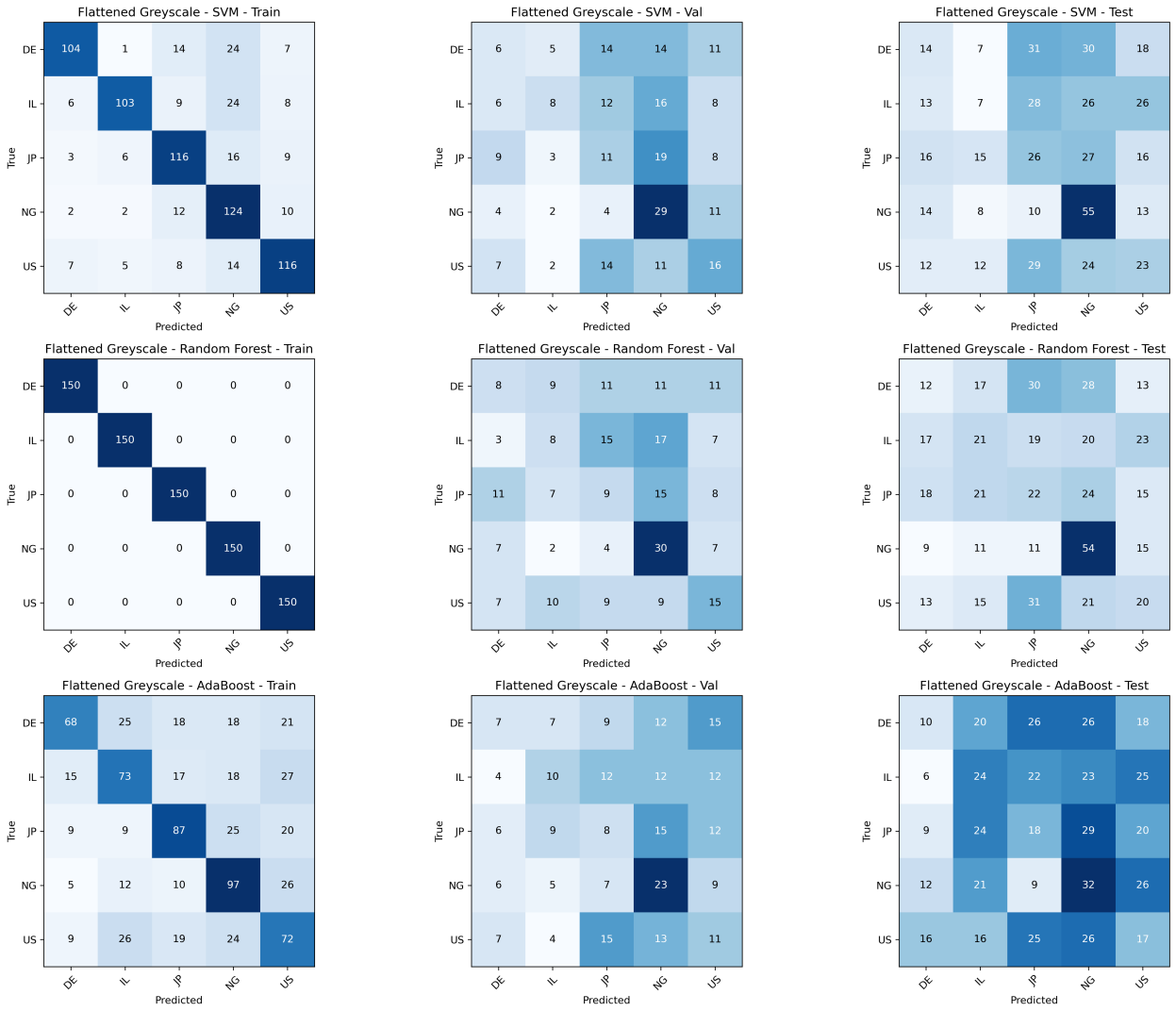


Figure 4: Flattened Grayscale confusion matrices on ML models.

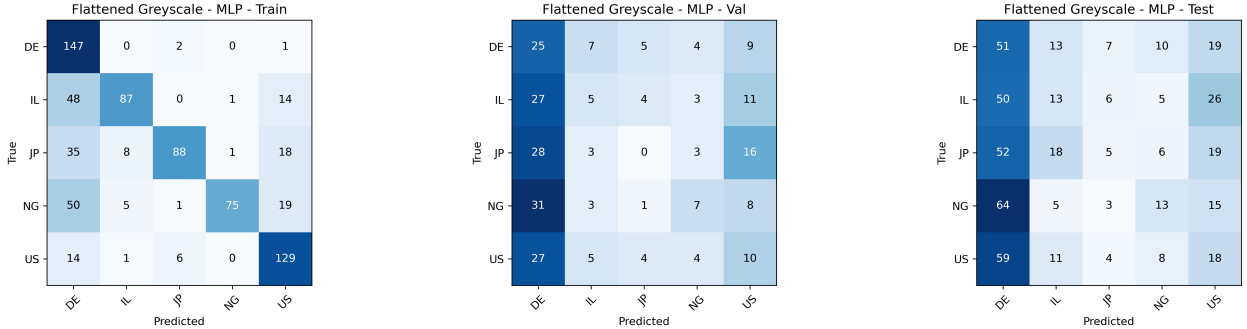


Figure 5: Flattened Grayscale confusion matrices on MLP model.

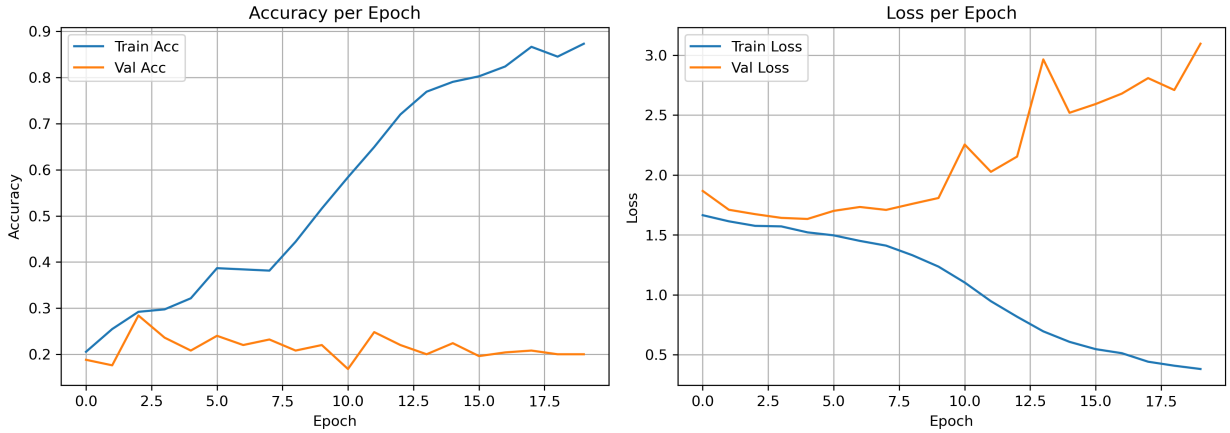


Figure 6: Flattened Grayscale MLP model training curves.

VGG Embeddings

The best VGG model is the MLP model with 33% accuracy despite overfitting. With the second best being SVM with 31% accuracy.

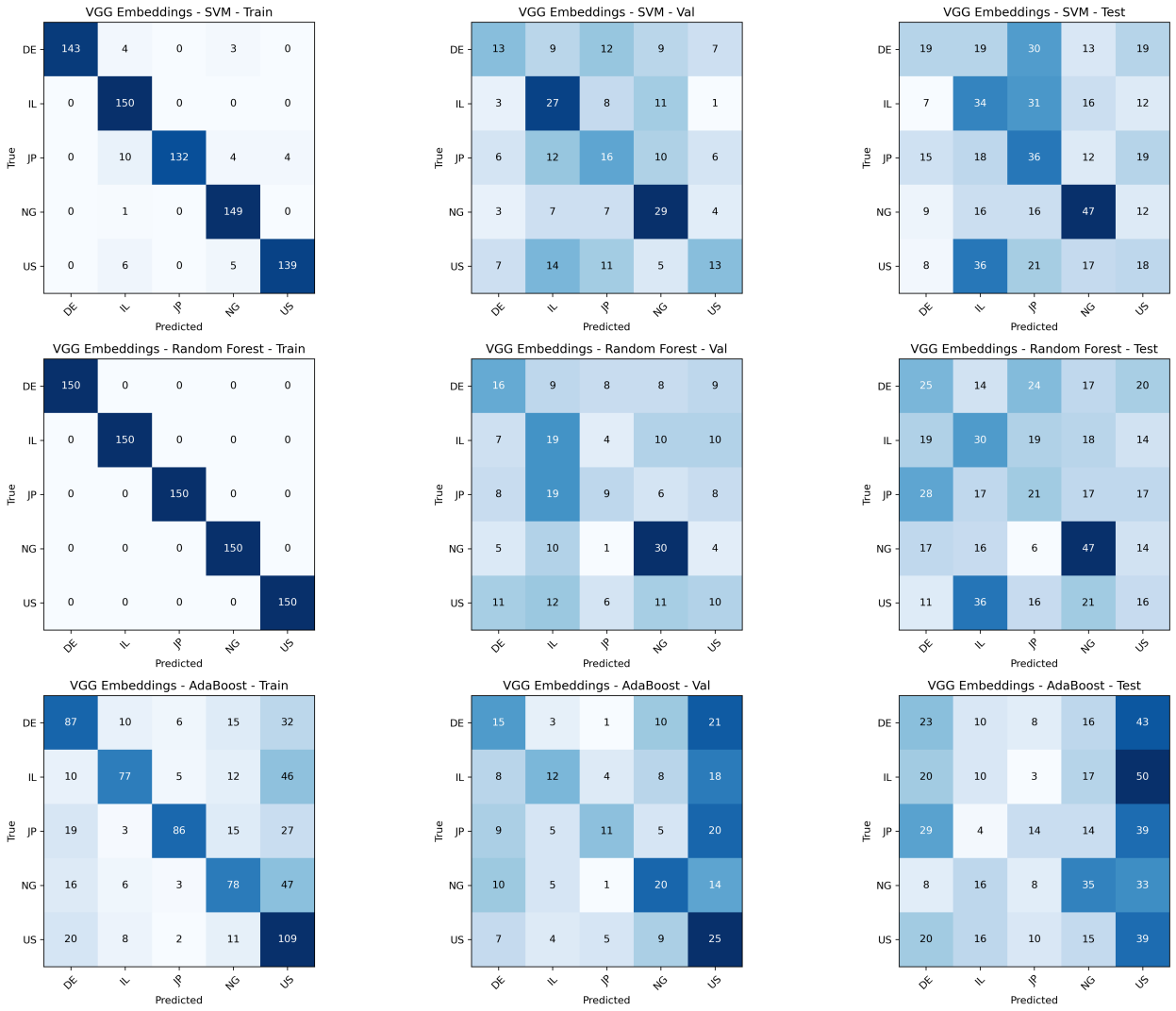


Figure 7: VGG Embeddings confusion matrices on ML models.

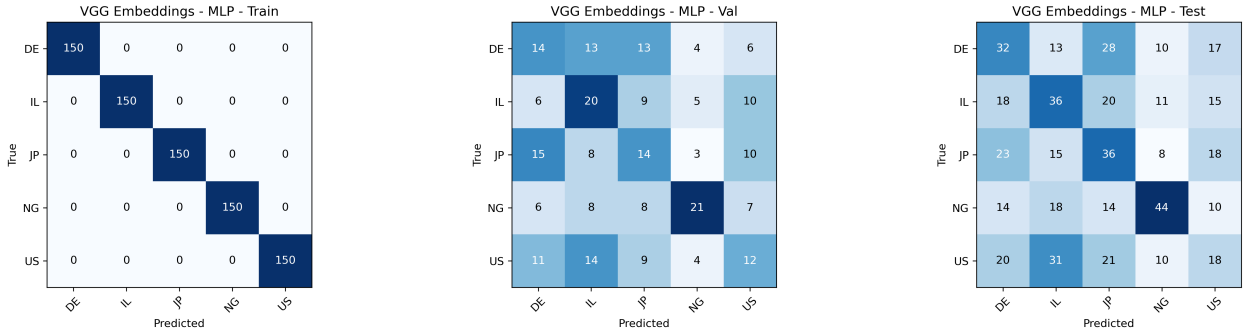


Figure 8: VGG Embeddings confusion matrices on MLP model.

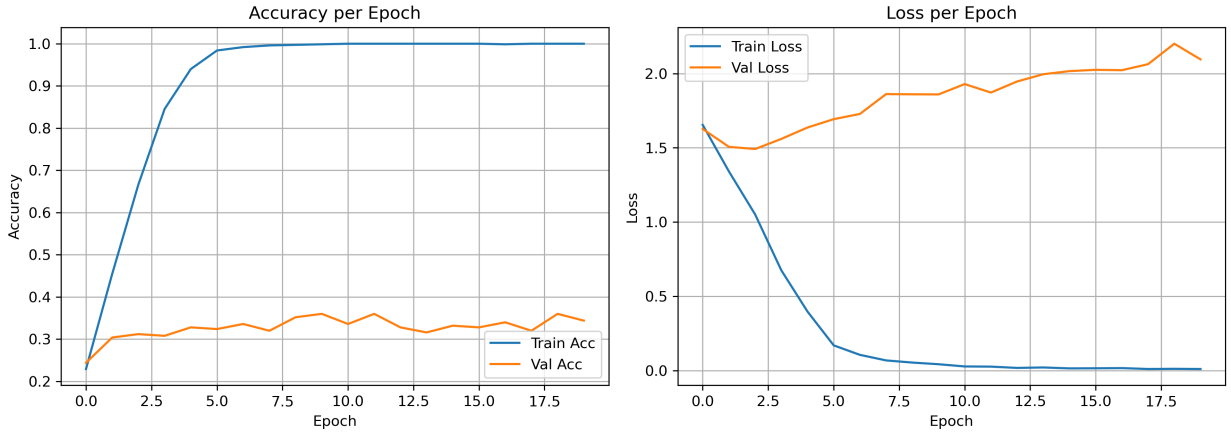


Figure 9: VGG Embeddings MLP model training curves.

Image Data

Training the image data itself involved fine-tuning the ResNet-50 model, by letting it retrain its parameters on our new data and with the 5 classes at the end of the model. It was trained for 20 epochs, with a learning rate of 10^{-3} and with l2 regularization with weight decay of 10^{-3}.

The results show a test accuracy of 28% but with the big advantage the the training data now seems to be generalized and not overfit, against the vectorized models which all were overfitting. However, a strange observation shows that the model biased its results across IL and NG labels.

Figure 10: Images confusion matrices on ResNet-50 model.

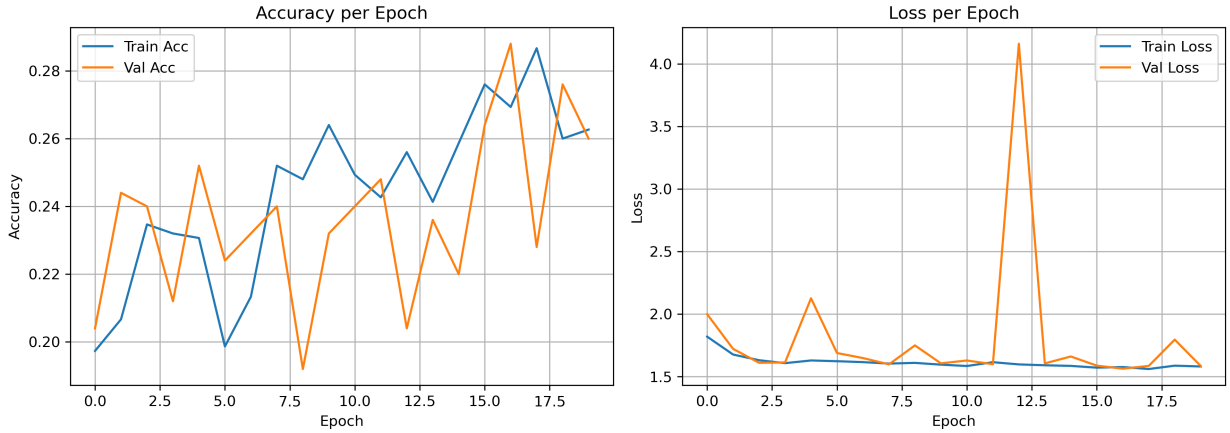


Figure 11: Images training curves.

Results Table

Method	Train Accuracy	Validation Accuracy	Test Accuracy	Precision	Recall	F1-Score	Train Time (s)
Flattened RGB - SVM	0.81	0.26	0.26	0.23	0.26	0.23	17.75
Flattened RGB - Random Forest	1	0.29	0.24	0.22	0.24	0.22	6.12
Flattened RGB - AdaBoost	0.56	0.22	0.24	0.23	0.24	0.23	199.74
Flattened RGB - MLP-49152	0.9	0.2	0.2	0.19	0.2	0.18	41.81
Flattened Greyscale - SVM	0.75	0.28	0.25	0.23	0.25	0.23	12.17
Flattened Greyscale - Random Forest	1	0.28	0.26	0.24	0.26	0.25	3.64
Flattened Greyscale - AdaBoost	0.53	0.24	0.2	0.2	0.2	0.2	60.97
Flattened Greyscale - MLP-16384	0.87	0.2	0.2	0.22	0.2	0.18	16.04
VGG Embeddings - SVM	0.95	0.39	0.31	0.31	0.31	0.3	14.58
VGG Embeddings - Random Forest	1	0.34	0.28	0.27	0.28	0.27	2.11
VGG Embeddings - AdaBoost	0.58	0.33	0.24	0.26	0.24	0.23	42.14
VGG Embeddings - MLP-25088	1	0.34	0.33	0.34	0.33	0.33	20.49
Images Dataset - ResNet50-5 class	0.26	0.26	0.28	0.27	0.28	0.21	194.3

The final results show that the best model in terms of test and validation scores is the MLP model of VGG Embeddings, with an accuracy score of 33%. The second best model, which is an ML model, is the SVM model of VGG Embeddings, with test accuracy score of 31%.
In total, it can be concluded that VGG-Embeddings performed the best overall across models, and that ResNet-50 Generalized the best, since it is the only model which doesn't overfit.
In addition, the results show that across the datasets, Adaboost seems to perform the worst with longest training time overall, and Random Forest always trains on 100% train and overfits the most.
In colclusion, we can infer that the best models for vectorized data are SVM and MLP, and that from all the datasets, the image data and embedding data perform the best.

Conclusions

- The **MLP trained on VGG embeddings** achieved the highest test accuracy (**33 %**), confirming that semantically rich features are more informative than raw pixels.
- **SVM on VGG embeddings** followed closely (31%), indicating that even shallow models benefit substantially from high-level representations.

- **ResNet-50 fine-tuning** scored a modest 28%, yet it was the *only* model that showed little overfitting, signalling better generalisation in spite of lower headline accuracy.
- **Flattened RGB & Grayscale vectors** led to severe overfitting; AdaBoost and MLP on these representations performed worst, while SVM provided the least-bad results (still ¡ 27%).
- Across all datasets, **AdaBoost** under-performed and required long training times; **Random Forest** memorised the training set (100% train accuracy) but failed to generalise.
- Overall, *embedding-based* and *image-based* pipelines outperform pure pixel vectors, and SVM/MLP emerge as the strongest choices for vectorised inputs.

Overall insight. High-level CNN embeddings compress class-relevant semantics that classical ML algorithms can harness, whereas direct pixel vectors swamp models with noise and colour variance. Fine-tuning an end-to-end CNN (ResNet-50) trades absolute accuracy for superior generalisation and robustness to label imbalance.

Challenges

- *Dimensionality explosion.* Flattened images produced vectors of up to 49152 features, inflating memory consumption and degrading distance-based learning. **Mitigation:** Used PCA for exploratory sanity-checks, and switched to 25088-D VGG embeddings for downstream modelling.
- *Severe overfitting.* Small sample size (few hundred images per class) led to almost perfect train accuracy yet poor test performance. **Mitigation:** Added BatchNorm, Dropout (0.3), ℓ_2 and weight decay (10^{-3}); still, embedding cue proved more decisive.
- *Compute constraints.* Training deep nets from scratch was infeasible on available GPU time. **Mitigation:** Adopted transfer learning (ResNet-50) and froze early layers for faster convergence; classical models ran on CPU realtively quickly.