OOP Design

I used 2 classes, namely, Perceptron and MultiClassPerceptron.

Perceptron is the binary classifier perceptron. It has the following functions: forward and learning rule in addition to the constructor and repr function. The constructor takes in 3 parameters, the number of inputs, the learning rate (alpha) and the weights initialization strategy. The initialization strategy could be zero, constant, uniform or gaussian. Usually uniform and gaussian help to break symmetry and we get a lower loss and better convergence with them.

The MultiClassPerceptron class uses 10 binary perceptrons and has 1 instance variable, perceptrons. This is an array or list of the 10 perceptrons, each one will belong solely to a distinct character, like perceptrons[0] will be the binary classifier to Bart Simpson. We have the multi_learning_rule function which identifies which perceptron is responsible for that input and corresponding output and sets the target to 1 if it is the correct binary perceptron or 0 otherwise. Then we run the learning rule on all the binary perceptrons. It also has a predict function, this returns the index of the maximum score. This is the function that will classify an input.

There are also 8 functions that are for evaluation metrics, per class and overall accuracy, per class and overall precision, per class and overall recall and per class and overall F1 Score. These functions return the per class and overall accuracy, precision, recall and F1 Score of our final model after hyperparameter tuning.

Data Normalization and Scaling

Before anything, we randomly shuffle the data, as it is in order after loading it into the respective NumPy arrays. Then we flatten each image as a vector. Then we apply one of 2 normalization techniques, minmax or gaussian/normal normalization. There is also an option of no normalization.

Training

We train the rgb and grayscale models separately both up to 100 epochs. There is an early stopping mechanism, if no improvement in accuracy is seen in the most recent 10 epochs, the loop stops and we take the 11th last model, as this will be our most accurate model during training. We also shuffle the training data each time for both RGB and grayscale, this ensures that the data is balanced, i.e. there’s, on average, at least a few of each class in the training data in each epoch.

We evaluate the model we train using a validation set, which is set at 10% of the overall data.

Hyperparameter Tuning

To achieve the best accuracy possible without taking randomness and shuffling into account, we must find the optimal hyperparameters.

I used gridsearch, and had combinations of the following hyperparameters:

learning_rates = [0.0001, 0.001, 0.01, 0.1]

init_strategies = ["zero", "constant", "uniform", "gaussian"]

normalization_techniques = ["minmax", "gaussian", None]

Using gridsearch on these hyperparameters, I found that the best RGB hyperparameters were,

RGB Top 1: LR=0.1, Init=uniform, Norm=minmax, Accuracy=0.4888

And for grayscale,

Grayscale Top 1: LR=0.0001, Init=constant, Norm=minmax, Accuracy=0.2875

*The Accuracy is measured on the validation set

The best RGB and grayscale model that achieved these respective accuracies were saved using the copy module.

RGB vs Grayscale

Below is a table of RGB against Grayscale for the overall evaluation metrics.

Model	Accuracy	Precision	Recall	F1
RGB (validation)	0.489	0.490	0.482	0.486
Grayscale (validation)	0.287	0.320	0.280	0.299
RGB (test)	0.431	0.425	0.431	0.428
Grayscale (test)	0.260	0.269	0.259	0.264

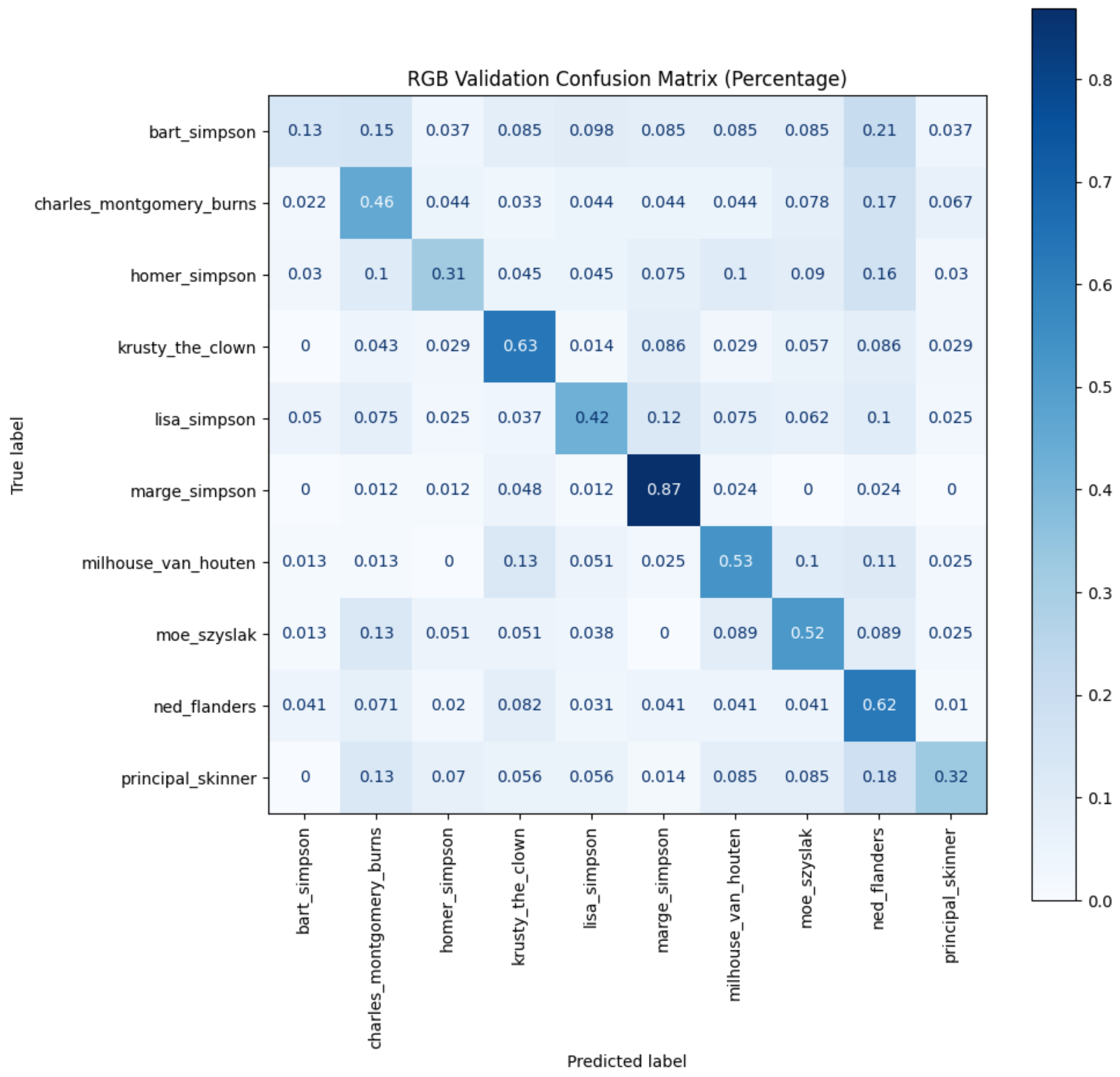
Clearly RGB outperforms grayscale in every overall metric.

However, grayscale does outperform RGB for some classes/characters. For example, Bart Simpson's RGB accuracy is 0.134, and in grayscale, Bart's accuracy is 0.451. Charles Burns also has a very slight increase in accuracy when using grayscale, RGB accuracy = 0.456 and grayscale accuracy = 0.478. Besides these 2 characters/classes, RGB greatly outperforms grayscale in all regards.

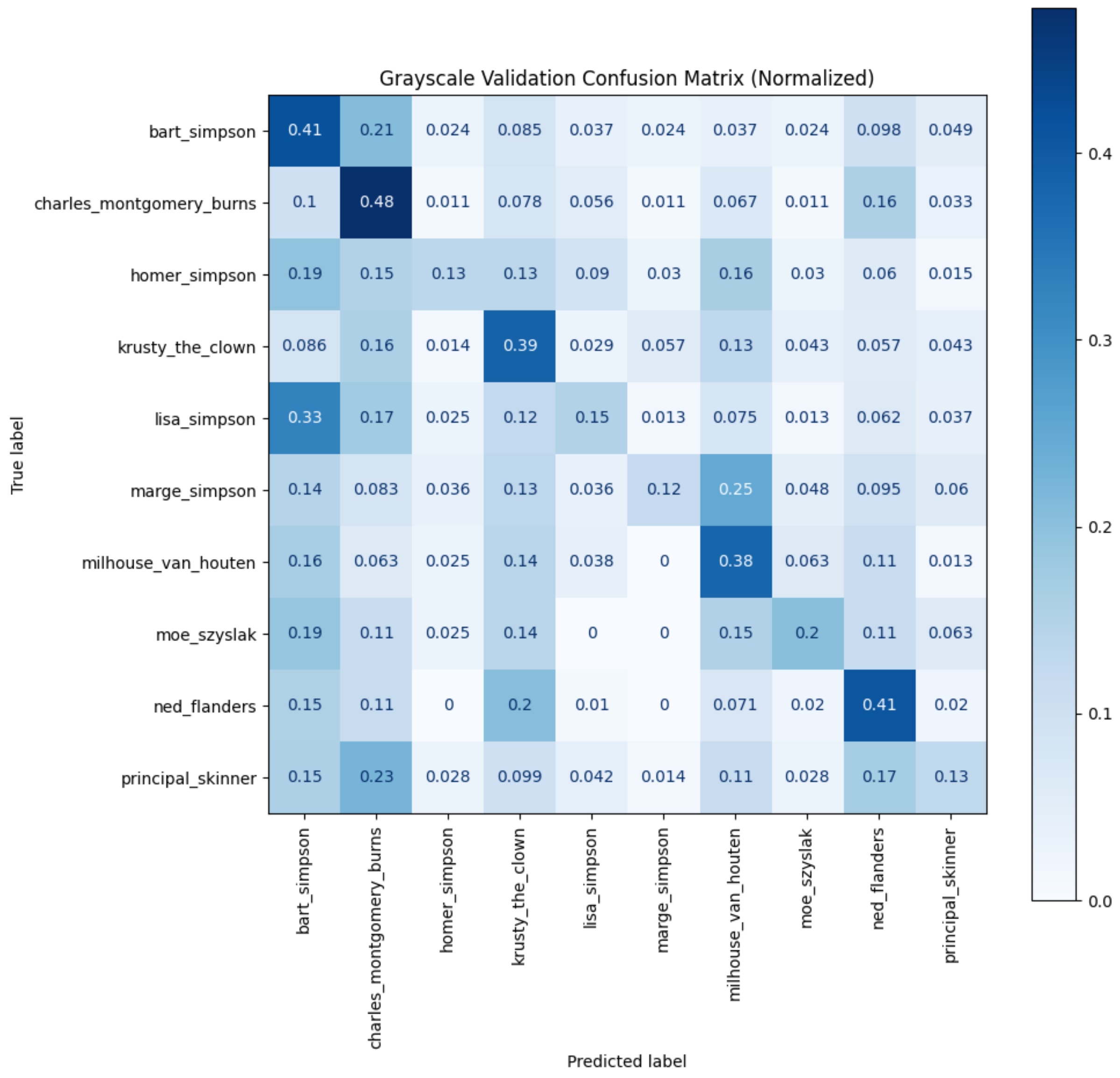
There are still characters that RGB struggles to identify, such as, again, Bart Simpson, Homer Simpson and Principal Skinner. These characters have sub-par accuracy.

Characters like Marge Simpson and Krusty the Clown get a huge accuracy improvement when using RGB, this is likely due to Marge's bright blue hair, and Krusty's off-white complexion in addition to his colourful hair.

Below are confusion matrices representing the accuracy on the validation set in percentages.



	bart_simpson	charles_montgomery_burns	homer_simpson	krusty_the_clown	lisa_simpson	marge_simpson	milhouse_van_houten	moe_szyslak	ned_flanders	principal_skinner
bart_simpson	0.13	0.15	0.037	0.085	0.098	0.085	0.085	0.085	0.21	0.037
charles_montgomery_burns	0.022	0.46	0.044	0.033	0.044	0.044	0.044	0.078	0.17	0.067
homer_simpson	0.03	0.1	0.31	0.045	0.045	0.075	0.1	0.09	0.16	0.03
krusty_the_clown	0	0.043	0.029	0.63	0.014	0.086	0.029	0.057	0.086	0.029
lisa_simpson	0.05	0.075	0.025	0.037	0.42	0.12	0.075	0.062	0.1	0.025
marge_simpson	0	0.012	0.012	0.048	0.012	0.87	0.024	0	0.024	0
milhouse_van_houten	0.013	0.013	0	0.13	0.051	0.025	0.53	0.1	0.11	0.025
moe_szyslak	0.013	0.13	0.051	0.051	0.038	0	0.089	0.52	0.089	0.025
ned_flanders	0.041	0.071	0.02	0.082	0.031	0.041	0.041	0.041	0.62	0.01
principal_skinner	0	0.13	0.07	0.056	0.056	0.014	0.085	0.085	0.18	0.32



	bart_simpson	charles_montgomery_burns	homer_simpson	krusty_the_clown	lisa_simpson	marge_simpson	milhouse_van_houten	moe_szyslak	ned_flanders	principal_skinner
bart_simpson	0.41	0.21	0.024	0.085	0.037	0.024	0.037	0.024	0.098	0.049
charles_montgomery_burns	0.1	0.48	0.011	0.078	0.056	0.011	0.067	0.011	0.16	0.033
homer_simpson	0.19	0.15	0.13	0.13	0.09	0.03	0.16	0.03	0.06	0.015
krusty_the_clown	0.086	0.16	0.014	0.39	0.029	0.057	0.13	0.043	0.057	0.043
lisa_simpson	0.33	0.17	0.025	0.12	0.15	0.013	0.075	0.013	0.062	0.037
marge_simpson	0.14	0.083	0.036	0.13	0.036	0.12	0.25	0.048	0.095	0.06
milhouse_van_houten	0.16	0.063	0.025	0.14	0.038	0	0.38	0.063	0.11	0.013
moe_szyslak	0.19	0.11	0.025	0.14	0	0	0.15	0.2	0.11	0.063
ned_flanders	0.15	0.11	0	0.2	0.01	0	0.071	0.02	0.41	0.02
principal_skinner	0.15	0.23	0.028	0.099	0.042	0.014	0.11	0.028	0.17	0.13

Ultimately, grayscale does not suffice to accurately identify almost all the characters. It is particularly good for Bart and Charles Burns, but other than that, RGB should always be used as the classifier.

An optimal approach would be to predict both the RGB and grayscale class and if grayscale predicts Bart or Charles Burns and the RGB predicts a low-accuracy class, such as Homer, then the grayscale prediction is more likely to be correct. Then obviously if both predictions match, then it is probably that common class. Otherwise just use the RGB prediction for the rest of the cases.

Conclusion

RGB has a big advantage over grayscale in almost every class, and it should be used instead of grayscale if a combination of the 2 is not feasible. Most characters are accurately identified by RGB as they have some form of distinct colour. This is what gives RGB that advantage.