

## Introduction

**Goal:** The aim of this study is to analyse machine learning methods in identifying and classifying blood cell images into one of eight classes. Four algorithms are tested for appropriateness with performance measured by compiling speed, accuracy, and error rates. Our goal is to understand the benefits and drawbacks of each model, recommend a model which presents good balance of all performance metrics, and explain our recommendation.

**Importance:** Pathology techniques (such as blood analysis) are used in every phase of healthcare provision, ranging from patient care (e.g., disease discovery, diagnosis, treatment, and monitoring) to research (e.g., disease prediction, prevention, and overall study) ([The Centre for International Economics, 2019](#)). Reliable, fast, and cost-efficient blood image identification solutions would increase healthcare accessibility to the general patient population and assist healthcare workers in making well-informed decision. To put things in perspective, “2 billion blood tests are performed in the United States each year, influencing 80% of medical decisions made in hospital and primary care settings” ([Balter, et al., 2018](#)).

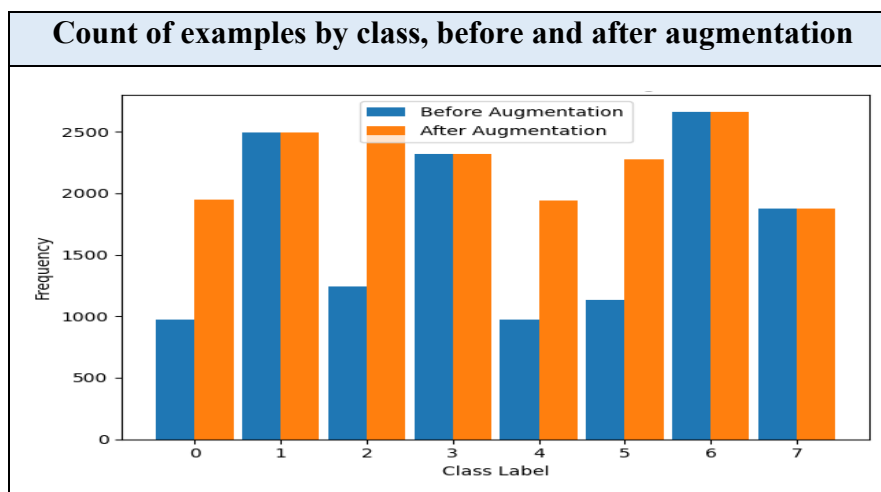
## Data description and exploration

**Original dataset:** The original dataset used for this study was produced by Acevedo, et al. from the Biomedical Diagnostic Center in Spain. The data contains 17,092 360x362 pixelated images, labelled into 8 different blood type categories. These images were annotated by expert clinical pathologists and “sourced from individuals without infection, hematologic or oncologic disease and free of any pharmacologic treatment at the moment of collection” ([Acevedo, et al., 2020](#)).









































**Dataset provided for this study:** the original images were (i) sized down into 28x28 pixelated images, (ii) split into two sets of arrays: X for each of the three red, green, and blue (RGB) values and y for class labels between 0 and 7 inclusively, and (ii) further split into train and test datasets with an 80:20 split. This processed dataset was done by Yang, et.al. and is licensed under [CC BY 4.0](#). Attribution provided via this link: ([Yang, et. al., 2023](#)).

### Initial data analysis results and possible challenges:

1. **Class distribution:** as shown in the histogram below, number of samples among classes ranges from c.1,000 for class 0 to c.2,600 for class 6. Data imbalanced may introduce bias to the model and cause negative effects on predictive performance ([Zheng & Jin, 2020](#)). Data Augmentation was also applied to balance out the classes reducing the difference in minimum and maximum class size by nearly 1000. Please see data distribution before and after augmentation below:

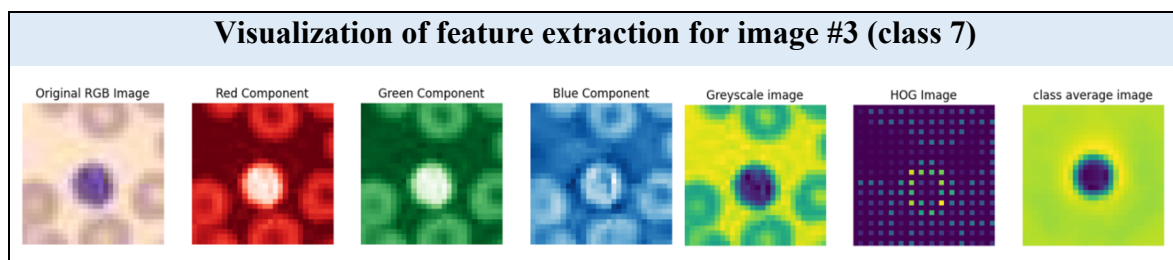


2. Class association: the provided images for this study were not labelled with their actual class names. The project team has referred to the data authors' whitepaper for guidance on actual class association (Acevedo, Alferez, Merino, Puigvi, & Rodellar, 2019):

Sample images from provided dataset						Class association and description <sup>a</sup>
Class 0						<b>Lymphocytes:</b> abundant and dark staining, condensed chromatin, scarce cytoplasm.
Class 1						<b>Eosinophils:</b> Bi-lobed nucleus, pink stained in cytoplasm granules, chromatin condensed and clumped.
Class 2						<b>Erythroblasts:</b> small (same size as red blood cells), picnotic chromatine.
Class 3						<b>Monocytes:</b> Kidney bean shaped nucleus; large and eccentrically placed. Abundant cytoplasm and presents some pink granules.
Class 4						<b>Basophils:</b> Bi-lobed nucleus with deep purple basophilic granules.
Class 5						<b>Immature granulocytes:</b> premature granulocytes which include promyelocytes, myelocytes, and metamyelocytes.
Class 6						<b>Neutrophils:</b> U-shaped nucleus, chromatin condensed, coarse, and clumped, moderate cytoplasm.
Class 7						<b>Platelets/Thrombocytes:</b> plate shaped, cytoplasmic fragments without nucleus.

**Notes:** <sup>a</sup>sourced from Acevedo, A., Alf  rez, S., Merino, A., Puigv  , L., & Rodellar, J. (2019). Recognition of peripheral blood cell images using convolutional neural networks. *Computer Methods and Programs in Biomedicine*, 180, 105020. <https://doi.org/10.1016/j.cmpb.2019.105020>

3. Feature detection: depending on the machine learning method used, several feature extraction techniques may be applied. Information on each color values, color intensities, and averages can be calculated and 'fed' into each machine learning algorithms for model fitting and prediction. Please see example below:



4. Possible challenges with dataset are listed below:

Challenges	Discussion on possible solutions
Data imbalance may introduce bias to the constructed model (Zheng & Jin, 2020).	<ul style="list-style-type: none"> <li>Under-sampling / over-sampling techniques may be applied (Google, 2023).</li> </ul>
Images has been scaled down which may result in loss of data granularity for modelling.	<ul style="list-style-type: none"> <li>Additional features may be created through techniques through different techniques such as the image data generator function from Tensorflow (Keras, 2023), histogram of oriented gradients (“HOG”) (Scikit-image, 2023), among others.</li> <li>Most salient features may also be extracted through principal component analysis (“PCA”).</li> </ul>
Images may not be centered perfectly as cells and nucleus are not perfectly round or have odd shapes.	
Some classes may have similar features to each other.	
Non-blood-cell matter presents noise in the images provided.	

5. Pre-processing: each algorithm requires different pre-processing as explained below. In general, neural network algorithms require less pre-processing as features are automatically created throughout the layers.

Pre-processing techniques and usage on each algorithm	FCNN <sup>a</sup>	CNN <sup>b</sup>	RF <sup>c</sup>	SVM <sup>d</sup>
<b>Min-max scaling:</b> RGB values are divided by its maximum value of 255 to create arrays of values between 0 and 1 to provide consistent scale and remove bias.	Yes	Yes	Yes	Yes
<b>Create additional features:</b> For example, imageDataGenerator from Keras creates new variation of images through rotation, standardization, shifts, flips, brightness changes, etc (Keras, 2023). HOG compute the gradient image in row and columns which ‘highlights’ edges/borders (Scikit-image, 2023).		Yes	Yes	Yes
<b>Flattening:</b> arrays are flattened into a single vector for easier processing by the algorithm.			Yes	Yes
<b>Feature extraction with PCA:</b> features which best explain variance in data are chosen to reduce data complexity without losing too much detail. In SVM, we applied PCA as a final process to identify features which explain 95% of total variances. As a result, 1,555 out of 14,016 were chosen.				Yes

Notes: <sup>a</sup>FCNN: Fully Connected Neural Network. <sup>b</sup>CNN: Convolutional Neural Network. <sup>c</sup>RF: Random Forest. <sup>d</sup>SVM: Support Vector Machine.

## Algorithm methods

---

### 1. Fully Connected Neural Network (“FCNN”)

#### Theory:

- The design of FCNN mimics the function of the brain which is composed of a stack of layers of artificial neurons. FCNN receives raw data via the **input layer** which is then processed and transformed through a series of **hidden layers** and combined into a result at the **output layer**. This form of one-way information flow (from input to output) without loops or feedback is also known as a **feedforward neural network** (Geron, 2019).
- A feedforward neural network is trained through a technique known as **backpropagation**. It is the process of calculating and readjusting the network’s weights at every layer and neuron backwards (from the output layer back to the input layer) at every cycle (known as **epoch**), until an acceptable performance level is achieved (Geron, 2019).
- Neural network is suitable for tasks which involves high-volume pattern recognition, classification, and decision making such as **image recognition**, natural language processing, speech recognition, among others (IBM, 2023).

#### Strengths and weaknesses:

Strengths	Weaknesses
<ul style="list-style-type: none"><li>• Low requirement for pre-processing.</li><li>• Output is generally accurate given large dataset.</li><li>• Modern computational power and tools allow for ‘brute force’ method in ML.</li></ul>	<ul style="list-style-type: none"><li>• High complexity which results in limited interpretability, difficult to tune, and long runtimes in large datasets.</li><li>• Optimum parameters may only be local optima, instead of global optima.</li></ul>

#### Architecture and hyperparameters:

We decided to design a ‘vanilla’ FCNN with sequential structure. Our neural network model starts with an **input layer which flattens given array**, passes the data through **four hidden layers which applies the same activation function**, and consolidates the output through a **softmax output layer**. Parameters to be tuned are listed below:

Parameters and values	Decision to tune
<b>n_hidden_neurons</b> [200, 300, 400]	There needs to be enough neurons at each hidden layer to allow for important features to be captured and reach convergence.
<b>activation_function</b> [relu, sigmoid, tanh]	Activation function determines how data is processed at every level. <b>ReLU</b> is a piecewise linear function and computationally efficient. <b>Sigmoid</b> and <b>Tanh</b> are non-linear, which helps with capturing complex relationship in data, but is more computationally expensive (Geron, 2019).
<b>learning_rate</b> [0.001, 0.01, 0.1]	Learning rate controls the step sizes taken to adjust the model’s parameter. The right learning rate is critical for the model to reach convergence and is discovered through trial and error.

## 2. Convolutional Neural Network (“CNN”)

### Theory:

- Convolutional Neural Networks (CNNs) are deep, feed-forward artificial intelligence models known for their less intensive parameter demands due to weight sharing, efficient object identification through hierarchical feature learning, and seamless integration of classification with feature extraction. (Asopa, 2018)
- While traditional image classification focuses on pixel-based strategies, leveraging contextual details like image shape has proven more effective, a strength of Convolutional Neural Networks. CNNs, increasingly noted for their context-aware classification capabilities, comprise four main components: a convolution layer, a pooling layer, an activation function, and a fully connected layer, each contributing distinct functionalities to the network's operation. (Asopa, 2018)

### Strengths and weaknesses:

Strengths	Weaknesses
<ul style="list-style-type: none"> <li>• CNNs outperform other algorithms due to their ability to automatically learn spatial hierarchies in images.</li> <li>• Pooling layers make CNNs more robust to distortions and variations in images.</li> <li>• Translation invariance enables CNNs to recognize objects regardless of their position in an image</li> </ul>	<ul style="list-style-type: none"> <li>• CNNs have many parameters making them prone to overfitting.</li> <li>• Requires significant computational resources in the training phase.</li> <li>• CNNs are sensitive to initialization and optimization algorithm which can lead to suboptimal solutions.</li> </ul>

### Architecture and hyperparameters:

The CNN model chosen for this task has 2 convolution and pooling layers which is a common pattern for progressively reducing the spatial dimensions of the input data while increasing feature depth. The final ‘Dense’ layer has 10 units with a softmax activation that efficiently maps inputs to each of the 10 distinct classes. Parameters to be tuned are listed below:

Parameters and values	Decision to tune
<b>optimizer:</b> [adam, rmsprop]	Defines the strategy for updating network weights during training, with 'adam' and 'rmsprop' representing different stochastic optimization methods. Choice of optimizer can affect training time and quality.
<b>dropout_rate:</b> [0.1, 0.3, 0.5]	Proportion of nodes in the layer that are randomly ignored to reduce risk of dependence on any one node. Tuning will help find optimal balance between underfitting and overfitting.
<b>num_filters_1:</b> [32, 64] <b>num_filters_2:</b> [64, 128]	Determines the number of kernels in the convolutional layers. More filters mean the network can capture more information about the input but may result in overfitting. Tuning will help avoid this problem.
<b>kernel_size:</b> [3, 5]	Dictates the height and width of the convolutional layers with larger layers capturing more information and smaller layers focusing on local details. Tuning can help find the ideal balance between the two.
<b>learning_rate:</b> [0.01, 0.001]	Scales the size of the step that the optimizer takes and controls the convergence of the training process. Too high of a learning rate can cause

---

the model to overshoot the minimum and too low of a rate can result in getting stuck in local minima.

---

### 3. Random Forest (“RF”)

#### Theory and justification:

- RFs are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large (Breiman, 2001).
- RF was chosen for this task because of its robustness in handling high dimensional data which makes it perfect for classifying noisy images. The algorithm is an ensemble method which mitigates overfitting through bagging and feature randomness and enhances prediction accuracy.

#### Strengths and Weakness:

Strengths	Weaknesses
<ul style="list-style-type: none"><li>• High predictive accuracy because of averaging process.</li><li>• Robustness against overfitting because of aggregation across trees.</li><li>• Offers insights into most informative features.</li></ul>	<ul style="list-style-type: none"><li>• Model complexity and difficulty in interpretability arising from aggregating hundreds or thousands of trees.</li><li>• High training time and memory usage.</li></ul>

#### Architecture and hyperparameters:

The data initially shaped for a CNN is flattened for this Random Forest

Parameters and values	Decision to tune
<b>N_estimators</b> [50, 100, 200, 250]	Dictates the number of trees in the forest that will be built and combined. Higher estimators can reduce overfitting up to a certain point. Tuning can optimize benefits vs computational cost.
<b>Max_features</b> [sqrt, log2]	Determines the number of features to consider when looking for the best split. Different strategies affect the training time and quality
<b>Max_depth</b> [10, 20, 30]	Specifies the maximum depth of the individual trees in the random forest, influencing how deep each tree can grow during training. Tuning could find a balanced point where the trees are deep enough to capture critical data while also being able to generalize



#### 4. Support Vector Machine (“SVM”)

##### Theory:

- SVM is a machine learning algorithm which creates a **hyperplane** that best separates data points (e.g., maximizes the margin of separation) of different classes. The hyperplane can be solved/discovered efficiently using the **kernel trick**, whereby the dot product of a pair of vectors is calculated in a higher space, thus bypassing the need to compute the complex mathematical transformation of each vector from the original space.
- To ensure that SVM is able to generalize well with data that does not clearly fit the separation by hyperplane, we can relax the algorithm by applying **soft-margin**, which allows some degree of misclassification of data (e.g., data points lying on the wrong side of the hyperplane) while still maximizing the margin of separation.
- Generally, SVM is appropriate for image recognition if it is provided with the data of the right features to focus on. Steps taken to pre-process the data is explained below.

##### Strengths and weaknesses:

Strengths	Weaknesses
<ul style="list-style-type: none"> <li>• Strong theoretical foundation.</li> <li>• Able to find the global minimum/maximum.</li> <li>• Able to handle high dimensional data through matrix operation (kernel trick)</li> <li>• Robust to overfitting through soft-margin SVM (C).</li> </ul>	<ul style="list-style-type: none"> <li>• Requires adequate pre-processing/feature engineering to run efficiently.</li> <li>• SVM can only separates data into a binary classification. Applying SVM to a multi-class problem requires additional complexity.</li> <li>• Sensitive to the choice of kernel and hyperparameters.</li> <li>• Model interpretability may be limited in higher-dimensional data.</li> </ul>

##### Architecture and hyperparameters:

We use a standard SVM with pre-processed data which has been (i) combined with extracted feature generated through the histogram of oriented gradient (“HOG”) technique to “highlight” image borders, (ii) scaled to remove bias from large values, and (iii) screened through principal component analysis (“PCA”), keeping features which cumulatively explains 95% of variances.

Parameters to be tuned are listed below:

Parameters and values	Decision to tune
<b>C</b> [0.1, 1, 10]	C is the regularization parameter which controls the trade-off between maximizing the margin of the decision boundary (high C) and minimizing the classification error on the training data (low C). The right value for C prevents over-fitting / allow for good generalization.
<b>Kernel</b> [rbf, sigmoid, poly]	Kernel determines which kernel trick is used in calculating the decision boundary. The three kernels chosen are non-linear which allows the model to capture complex relationship in our image data. RBF (radial basis function) measures similarity between two data points by its

	Euclidian distance, Sigmoid by its hyperbolic tangent function, and polynomial by its polynomial features ( <a href="#">scikit-learn, 2023</a> ).
<b>Gamma</b> <b>[auto, scale]</b>	Gamma decides the curvature we apply on the hyperplane in the original space. High gamma value means only points closest to the hyperplane will carry the weight, leading to a smoother boundary ( <a href="#">scikit-learn, 2023</a> ): <ul style="list-style-type: none"> <li>• Gamma = auto sets gamma to (1/number of features)</li> <li>• Gamma = scale sets gamma to (1/(number of features * X.var()))</li> </ul>

## Results and discussions

Notes on results: Results are presented in table format below for conciseness. It should be noted that the team was only able to utilize multi-core function (n\_jobs = -1) in GridSearchCV for random forest and support vector machine. As such, runtimes should only be used to compare computational load within the same model, but not across models.

### 1. Fully Connected Neural Network:

- **Accuracy score** is more stable for models with activation\_function = “**tanh**”. Output of “tanh” is centred around 0 and ranges between -1 and 1, thus preventing the vanishing gradient problem.
- **Runtimes** are generally higher with smaller learning rates but can be unpredictable for different number of hidden neurons. This may be caused by different rates taken by each model to reach convergence. Some simpler models take longer time to reach convergence while models with higher complexity (e.g., higher number of hidden neurons) may reach convergence faster).
- Hyperparameter tuning results shown below:

		Mean test score			Mean fit times (seconds)		
Activation function		relu	sigmoid	tanh	relu	sigmoid	tanh
Hidden neurons	Learning rate						
200	0.001	0.747	0.194	0.792	78.0	293.8	774.7
	0.010	0.144	0.449	0.817	77.9	85.8	88.6
	0.100	0.071	0.737	0.747	81.9	86.3	89.1
300	0.001	0.752	0.192	0.746	88.8	87.7	98.5
	0.010	0.071	0.467	0.830	85.7	408.0	100.7
	0.100	0.071	0.724	0.798	84.6	665.5	91.8
400	0.001	0.740	0.195	0.786	91.2	103.3	97.2
	0.010	0.071	0.502	0.825	90.8	92.9	95.2
	0.100	0.071	0.702	0.756	92.4	89.0	92.1

Note: test scores below 0.7 is highlighted with red font. Best test score is highlighted in green.

### 2. Convolutional Neural Network

- **Accuracy score** was remarkably consistent with the score never dipping below 0.820 on any combination of hyperparameters suggesting that the Convolutional Neural Networks implementation was well-designed and robust.
- **Runtimes** were consistent across different hyperparameters with a difference of only 27 seconds between maximum and minimum runtimes. As expected, the runtime



## COMP5318 – Assignment 2

Report number: A2-Report- 530653709-530454874

consistently increased as learning rate decreased and as kernel size, dropout rate and number of filters increased. There was no significant difference between the runtimes of the optimizers **adam** and **rmsprop**.

- Abridged hyperparameter tuning results shown below, please see appendix for full results:

Abridged results for: (n_filter_1 = 32, n_filter_2 = 128)			Mean test score		Mean fit times (seconds)	
Optimizer			adam	rmsprop	adam	rmsprop
Dropout rate	Kernel size	Learning rate				
0.1	3	0.001	0.839	0.843	24.7	24.4
		0.01	0.845	0.846	25.6	25.3
	5	0.001	0.854	0.853	29.4	29.5
		0.01	0.873	0.857	29.6	29.4
0.3	3	0.001	0.855	0.836	26.0	26.0
		0.01	0.823	0.848	25.8	25.8
	5	0.001	0.860	0.844	32.6	32.4
		0.01	0.839	0.840	30.2	30.2
0.5	3	0.001	0.845	0.855	27.6	27.9
		0.01	0.853	0.846	26.3	25.8
	5	0.001	0.844	0.862	40.7	38.7
		0.01	0.844	0.848	31.3	31.6

Note: test scores below 0.7 is highlighted with red font. Best test score is highlighted in green.

### 3. Random Forest

- **Accuracy score** is stable across all parameters, suggesting random forest model has no issues with convergence.
- **Runtimes** are consistently higher using max\_feature = “sqrt” compared to max\_feature = “log2”. As expected, the runtimes also get significantly longer as n\_estimators increases. Counterintuitively, at high n\_estimators count, an increase in max\_depth from 20 to 30 decreased mean fit times when ordinarily this would increase mean fit times. This may be happening because of early stopping due to a lack of splits or pruning branches that don’t provide sufficient information gain to a lesser depth.
- Hyperparameter tuning results shown below:

		Mean test score			Mean fit times (seconds)		
Max_depth		10	20	30	10	20	30
N_estimators	Max features						
50	log2	0.763	0.778	0.780	9.4	13.2	13.9
	sqrt	0.775	0.791	0.795	36.9	55.8	55.3
100	log2	0.769	0.794	0.795	18.2	25.8	26.4
	sqrt	0.776	0.798	0.806	74.1	106.4	111.1
200	log2	0.774	0.800	0.798	35.5	51.5	53.1
	sqrt	0.783	0.809	0.808	142.7	214.2	186.9
250	log2	0.774	0.801	0.800	44.3	64.6	59.9
	sqrt	0.783	0.807	0.810	174.9	252.6	209.1

Note: test scores below 0.7 is highlighted with red font. Best test score is highlighted in green.

### 4. Support Vector Machine

## COMP5318 – Assignment 2

Report number: A2-Report- 530653709-530454874

- **Accuracy score** is generally more stable for kernel = **rbf**. The rbf kernel is suitable for high dimensional data and pattern recognition.
- **Runtimes** are generally higher when gamma = auto (which results in lower gamma value) and lower C values (which lowers the effects of regularization), both making it difficult to find a decision boundary.
- Hyperparameter tuning results shown below:

		Mean test score			Mean fit times (seconds)		
Kernel		rbf	sigmoid	poly	rbf	sigmoid	poly
Gamma	C						
scale	0.1	0.712	0.740	0.490	160.6	123.0	246.4
	1	0.858	0.785	0.752	117.6	70.2	236.9
	10	0.875	0.726	0.801	142.6	49.8	268.8
auto	0.1	0.655	0.588	0.195	186.1	206.8	275.1
	1	0.795	0.755	0.195	107.7	120.6	296.2
	10	0.857	0.837	0.322	75.6	77.1	202.6

Note: test scores below 0.7 is highlighted with red font. Best test score is highlighted in green.

### 5. Best model comparison

- As displayed below, **Convolutional Neural Network** performed best on every metric for most classes.
- We note that the performance varies among classes due to similarities / fuzziness. For example, class 0 has lower precision, recall, and f1-scores possibly due to its features being 'too generic' or without obvious differentiating features and class 7 performed best possibly because it looks most different to any other classes.

Scores on test data												
Best model parameters and scores	FCNN			CNN			RF			SVM		
	hidden neurons: <b>300</b> learning rate: <b>0.01</b> activation: <b>tanh</b>			optimizer: <b>rmspop</b>			max depth: <b>30</b> max features: <b>sqrt</b> n estimators: <b>250</b>			C: <b>10</b> , gamma: <b>scale</b> kernel: <b>rbf</b>		
				dropout rate: <b>0.3</b>								
				num filter 1: <b>64</b>								
				num filter 2: <b>128</b>								
				kernel size: <b>5</b>								
			learning rate: <b>0.001</b>									
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Class 0	0.76	0.59	0.66	0.80	0.71	0.75	0.73	0.61	0.66	0.77	0.72	0.74
Class 1	0.91	0.97	0.94	0.97	0.98	0.98	0.92	0.92	0.92	0.96	0.95	0.96
Class 2	0.96	0.73	0.83	0.98	0.87	0.92	0.90	0.86	0.88	0.93	0.85	0.89
Class 3	0.70	0.65	0.67	0.71	0.87	0.78	0.69	0.73	0.71	0.75	0.83	0.79
Class 4	0.68	0.86	0.76	0.94	0.84	0.88	0.79	0.85	0.82	0.82	0.80	0.81
Class 5	0.74	0.65	0.69	0.88	0.65	0.75	0.67	0.63	0.65	0.83	0.76	0.79
Class 6	0.84	0.95	0.89	0.94	0.97	0.96	0.90	0.93	0.91	0.92	0.94	0.93
Class 7	0.98	1.00	0.99	1.00	1.00	1.00	1.00	0.99	1.00	0.99	0.99	0.99
Accuracy	-	-	0.83	-	-	0.90	-	-	0.84	-	-	0.88
Macro average	0.82	0.80	0.80	0.90	0.86	0.88	0.82	0.81	0.82	0.87	0.86	0.86
Weighted avg.	0.83	0.83	0.83	0.90	0.90	0.90	0.84	0.84	0.84	0.88	0.88	0.88

Note: test scores below 0.7 is highlighted with red font. Best test score is highlighted in green.

## 6. Conclusion

Taking run-times and accuracy scores into account, we recommend using a **Convolutional Neural Network** for noisy image classification tasks. To further expand this study, we recommend expanding the number of parameters tested, especially for neural network models, to ensure that we find the global optimum solution and explore additional pre-processing techniques for random forest and support vector machine to ensure data provided to the model provides the best information on features. Specific recommendations are listed below:

- **Fully connected neural network:** different designs of neural network, including different types of input layers, different types and number of hidden layers, and different types of output layers.
- **Convolutional neural network:** altering the architecture by increasing the number of convolution and pooling layers, additional hyperparameter values, including more optimizer algorithms and a finer array of learning rates.
- **Random forest:** significantly expanding the number and range of hyperparameters tested during grid search such as `class_weight`, `max_leaf_nodes`, `bootstrap` and `min_samples_leaf`
- **Support vector machine:** tweaking `class_weight` (e.g. giving more weight to less frequent classes or classes that perform worse than others such as class 0), testing a finer array of C values.

## **Reflection**

---

Each algorithm has different use cases and trade-offs which should be considered. The choice of algorithm must be matched with the goal of the exercise itself. More complex algorithm may take longer to run and may not provide the best explanation/reasoning on feature importance. Users need to consider what output needs to be generated to (i) measure the quality of the model and (ii) provide useful insights when designing a model protocol. Simpler models may provide adequate outputs with lower computational costs (e.g., shorter runtimes) given the right pre-processing and design.

**COMP5318 – Assignment 2**  
**Report number: A2-Report- 530653709-530454874**

**Appendix – CNN results:**

Dropout rate	Num Filters 1	Num Filters 2	Kernel size	Learning rate	Mean test score by optimizer		Mean fit time by optimizer	
					adam	rmsprop	adam	rmsprop
0.1	32	64	3	0.001	0.843	0.834	24.7	25.1
				0.01	0.833	0.832	26.9	24.9
			5	0.001	0.844	0.835	27.1	27.0
				0.01	0.839	0.845	27.2	26.6
		128	3	0.001	0.839	0.843	24.7	24.4
				0.01	0.845	0.846	25.6	25.3
			5	0.001	0.854	0.853	29.4	29.5
				0.01	0.873	0.857	29.6	29.4
	64	64	3	0.001	0.819	0.844	38.4	38.9
				0.01	0.847	0.836	42.3	43.1
			5	0.001	0.855	0.847	38.8	39.0
				0.01	0.834	0.848	37.8	38.3
		128	3	0.001	0.848	0.866	41.1	41.5
				0.01	0.859	0.866	45.9	45.3
			5	0.001	0.861	0.858	42.8	44.4
				0.01	0.851	0.863	44.4	43.7
0.3	32	64	3	0.001	0.841	0.820	25.7	26.1
				0.01	0.832	0.835	26.4	26.4
			5	0.001	0.839	0.849	28.6	29.2
				0.01	0.833	0.824	28.4	28.1
		128	3	0.001	0.855	0.836	26.0	26.0
				0.01	0.823	0.848	25.8	25.8
			5	0.001	0.860	0.844	32.6	32.4
				0.01	0.839	0.840	30.2	30.2
	64	64	3	0.001	0.847	0.833	40.1	39.4
				0.01	0.850	0.841	39.8	39.1
			5	0.001	0.820	0.850	40.6	42.2
				0.01	0.836	0.840	39.1	40.8
		128	3	0.001	0.851	0.850	43.2	42.7
				0.01	0.856	0.844	42.3	43.0
			5	0.001	0.864	0.859	44.8	46.3
				0.01	0.858	0.841	44.7	44.8
0.5	32	64	3	0.001	0.826	0.842	25.5	25.9
				0.01	0.830	0.842	25.5	25.5
			5	0.001	0.829	0.837	29.9	32.5
				0.01	0.839	0.836	28.5	28.6
		128	3	0.001	0.845	0.855	27.6	27.9
				0.01	0.853	0.846	26.3	25.8
			5	0.001	0.844	0.862	40.7	38.7
				0.01	0.844	0.848	31.3	31.6
	64	64	3	0.001	0.840	0.834	41.3	40.3
				0.01	0.845	0.832	39.7	39.7
			5	0.001	0.841	0.842	54.6	52.1
				0.01	0.834	0.836	39.0	39.1
		128	3	0.001	0.860	0.844	43.7	43.3
				0.01	0.859	0.863	42.6	42.5
			5	0.001	0.868	0.852	55.7	56.1
				0.01	0.848	0.848	44.4	47.2

## **References**

- Acevedo, A., Alferez, S., Merino, A., Puigvi, L., & Rodellar, J. (2019). Recognition of peripheral blood cell images using convolutional neural networks. *Computer Methods and Programs in Biomedicine*.
- Acevedo, A., Merino, A., Alferez, S., Molina, A., Boldu, L., & Rodellar, J. (2020). *A dataset for microscopic peripheral blood cell images for development of automatic recognition systems*. Mendeley Data.
- Asopa, S. I. (2018). Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. *Procedia Computer Science*, 679-688.
- Balter, M. L., Leipheimer, J. M., Chen, A. I., Shrirao, A., Maguire, T. J., & Yarmush, M. L. (2018). Automated end-to-end blood testing at the point of care: Integration of robotic phlebotomy with downstream sample processing. *Technology Volume 6 Number 2 (World Scientific)*, 59-66.
- Breiman, L. (2001). Random Forests. *Machine Learning*.
- Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilley.
- Google. (2023). *Google Foundational Courses*. Retrieved from Google: <https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data>
- IBM. (2023). *IBM*. Retrieved from IBM: <https://www.ibm.com/topics/neural-networks>
- Keras. (2023). *Keras*. Retrieved from Keras: [https://keras.io/api/data\\_loading/image/](https://keras.io/api/data_loading/image/)
- Scikit-image. (2023). *Scikit-image*. Retrieved from Scikit-image: <https://scikit-image.org/docs/dev/api/skimage.feature.html#skimage.feature.hog>
- scikit-learn. (2023). *Plot classification boundaries with different SVM kernels*. Retrieved from scikit-learn: [https://scikit-learn.org/stable/auto\\_examples/svm/plot\\_svm\\_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-py](https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-py)
- scikit-learn. (2023). *RBF SVM Parameters*. Retrieved from scikit-learn: [https://scikit-learn.org/stable/auto\\_examples/svm/plot\\_rbf\\_parameters.html](https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html)
- Tensorflow. (2023). *Tensorflow*. Retrieved from Tensorflow: [https://www.tensorflow.org/api\\_docs/python/tf/keras/preprocessing/image/ImageDataGenerator](https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator)
- The Centre for International Economics. (2019). *The Economic Value of Pathology: Achieving Better Health, and a Better Use of Health Resources*. The Centre for International Economics.
- Yang, J., Shi, R., Liu, Z., Zhao, L., Ke, B., Pfister, H., & Ni, B. (2023). MedMNIST v2 - A large-scale lightweight benchmark for 2D and 3D biomedical image classification . *Scientific Data*.
- Zheng, W., & Jin, M. (2020). The Effects of Class Imbalance and Training Data Size on Classifier Learning: An Empirical Study. *SN Computer Science*.