# Evaluating the Performance of Multilayer Perceptrons and Support Vector Machines on an Image Classification Task

Gibbs-Bravo A., Pennacchia D.
*Neural Computing, MSc Data Science*
School of Mathematics, Computer Science and Engineering
City, University of London

March 2019

**Abstract**

This paper evaluates the prediction accuracy of a Multilayer Perceptron and a Support Vector Machine, in a multiclass classification task of image recognition using the widely-popular CIFAR-10 dataset. We show that both of these models are able to learn and classify images with a relatively high degree of accuracy. Hyperparameters were tuned using a grid-search to achieve the best possible performance for each model, whose classification rate was compared on a separate test sample using confusion matrices. SVM outperformed MLP providing a better classification rate using both CNN and Surf feature extractors while MLP achieved better performance on raw data.

***Keywords*—** Image Classification, SVM, MLP, Neural Computing

# 1 Introduction & Motivation

The dominant success of Geoff Hinton's lab on the ImageNet image classification competition in 2012 using a CNN architecture has been largely attributed as the cause of the resurgence of deep neural networks, particularly in computer vision [13]. CIFAR-10 is a similar data-set to ImageNet containing 60.000 color images with 32x32 pixel resolution of 10 classes of objects. The task is multi-class classification where the objective is to label the images correctly.

Advancements in image classification provide real-world applications such as healthcare and medical engineering (PET, MR, scan and computer-aided diagnosis) and in other industries which are invaluable [12].

The purpose of this paper is to perform our multi-class classification to compare Multilayer Perceptrons (MLP) and Support Vector Machines (SVM) based on the CIFAR-10 Dataset [2], with the aim to learn which performs better for image classification.

## 1.1 Hypothesis Statement

We expect that the MLP algorithm will perform better than SVM although it will require a greater focus on hyperparameter tuning and require further training time [11]. We also expect that feature extraction and representation is more important than the algorithm itself and that using a convolutional neural network as a feature extractor will result in the best performance [6, 7].

## 1.2 Multilayer Perceptron (MLP)

The multilayer perceptron (MLP) algorithm is a universal function approximator which was created to solve the original issues of the Perceptron algorithm which was unable to solve XOR problems [10]. It is a feedforward neural network used to extract patterns in supervised learning which consists of an input layer, one or more hidden layers, and an output layer. Each layer except the input layer contains nonlinear activation functions of which *TanH, ReLu*, and *Sigmoid* are currently the most common [15]. The weights of the nodes are updated as a result of two passes: the forward pass which applies the weights to the input to make a prediction and the backward pass which uses backpropagation to propagate the error across the weights and adjust them in the direction of their gradient to reduce the prediction error [10].

| Advantages | Disadvantages |
|---|---|
| Universal function approximation results in state of art performance in many task [4] | Computationally expensive [16] |
| Able to represent complex features without any manual engineering | Generally requires significant amounts of data for strong performance |
| Performs well when data has a high number of features[24] | Black box and lacks mathematical proofs explaining efficacy of deep networks [18] |

### 1.3 Support Vector Machine (SVM)

The Support Vector Machine algorithm was developed in Russia in the early 60's as a way of a nonlinear generalization of the Generalized Portrait Algorithm, applying the kernel trick, namely transforming data into another dimension that has a clear dividing margin between different classes. For classification, SVMs operate by finding a hypersurface in the space of possible inputs. This hypersurface will attempt to separate items belonging to the same class [5]. SVMs are inherently two-class classifiers, so when facing a multi-class problem the most common techniques are mainly two:

- Build a one-versus-all classifier (*OVA* classification), where for each of K-binary classifiers, one class is positive and the rest is negative.
- Build a one-versus-one classifiers (*OVO* classification), which trains K(K-1)/2 binary classifiers. For each binary learner, only two classes are compared (the rest are ignored), so the learner must learn to distinguish these two classes.

| Advantages | Disadvantages |
|---|---|
| Works well on smaller and cleaner datasets [20] | Less effective on noiser with overlapping classes datasets |
| Useful for both linearly separable and non-linearly separable data | Depending on the kernel used, computational time might be very slow |
| Memory efficient: SVM uses a subset of training points to make the final classification | There is no proper probabilistic interpretation of SVM. Probability estimates are calculated using cross-validation and could be inconsistent with the scores [22]. |

## 2 Dataset

CIFAR-10 is one of the most widely used datasets in computer vision [3], [1] and consists of 60.000 RGB images with 32x32 pixel resolution. The classes are evenly split with 6.000 images (5000 training and 1000 test images) from each label. The data is structured as a vector of three dimensional tensors representing the pixel intensity (between 0-255) across the three color channels (RGB). As this cannot be used directly by a classifier, we need to reshape the data and extract features.

We will compare the performance between tuned SVM and MLP classifiers on the following feature extraction methods:

- *Raw pixel intensity*: flattens the raw pixel intensity values and is expected to yield the lowest performance as the data lacks informative structure and raw data tends to be redundant [12]. Converts the tensor into matrix of 60.000 rows and 3.072 features (32x32x3).
- *Speeded up robust features (Surf)*: A feature detector and descriptor that uses an integer approximation of the determinant of Hessian matrix ('BLOB Detector') in order to find interest points (defined as salient features from a scale-invariant representation), local neighborhood descriptors (which are scale and orientation invariant) and matching pairs (comparison of images based on predetermined interest points and descriptors) [19]. Converts the tensor to matrix with the dimension of 60.000 rows and 500 features.
- *Convolutional Neural Network (CNN)*: applies a series of convolutional filters which are trained to extract features to classify images. [10]. One of the greatest benefits of CNNs is that models trained from similar images can be used to extract features on other images called transfer learning. We apply the pretrained VGG-16 network as a feature extractor. We freeze the convolutional layers of the CNN and only train the classifiers on the extracted feature vector. Converts the tensor to matrix with the dimensions of 60.000 rows and 4.096 features.

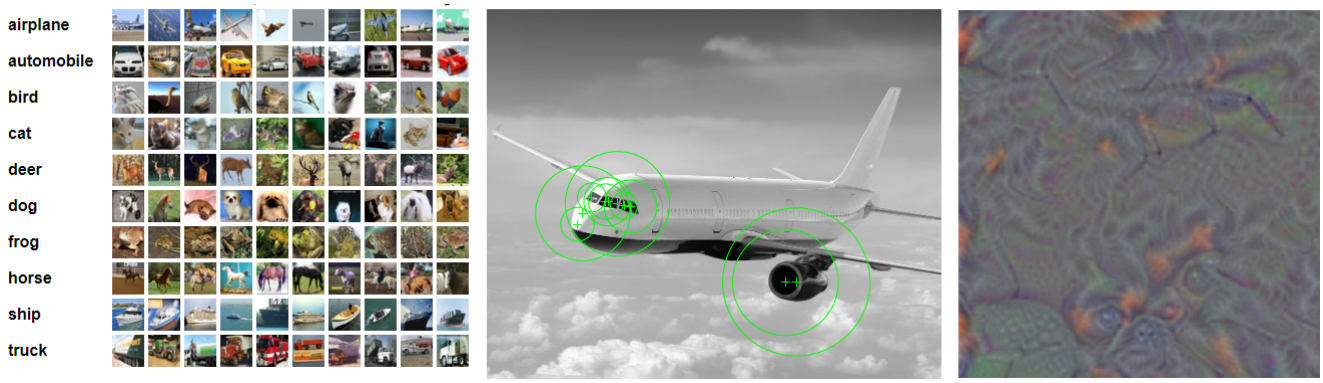


*Figure 1: Cifar-10 dataset: Raw pixels: sample of raw images from dataset (left), Surf: 10 strongest surf feature points on samples image (center), CNN: sample filter activation visualized using DeepDream (right)*

# 3 Methodology

The methodology we used to train, validate, and test the models consists of splitting the data into an initial train and holdout test set. For our experiments we only used 20.000 training images and 5.000 holdout test images to reduce computational time. We then split the training set into a further training set and a validation set used to grid-search model hyperparameters. Given the limited computational resources, we were unable to perform k-fold cross validation, therefore there may be variance in the validation set performance. Once the optimal hyperparameters are selected, each model is retrained on the full training dataset to increase the number of training samples and is evaluated on the holdout set (Figure 2).

In our experiments, we compare both models after conducting a grid-search of hyperparameters for each of the three feature extraction techniques in two stages to reduce the search space.

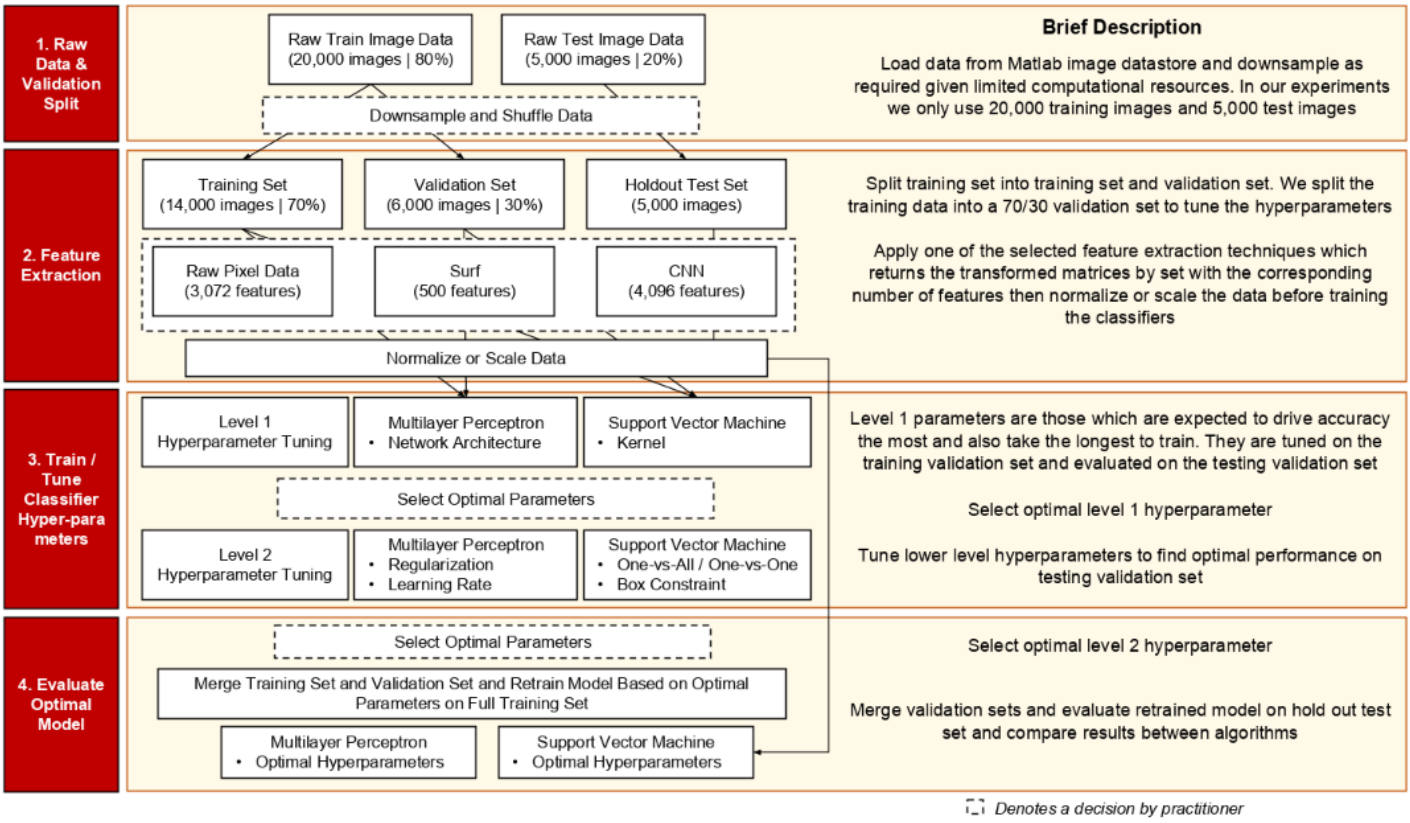


*Figure 2: Data-flow structure*

## 3.1 Architecture and Parameters used for the MLP

Given the problem is multiclass classification, a softmax classifier is used with a cross-entropy loss function [14].The optimal network architecture for the MLP is first gridsearched from the possible configurations below (Figure 3). The activations function for the each node is 'tansig' (tanh) which tends to outperform 'logsin' (sigmoid) [15] and gradient descent with momentum and adaptive learning rate backpropagation is used as the optimizer. The model training typically terminates using early stopping when the error on the validation set increases for 10 consecutive epochs. However, training is also capped at 250 epochs or 10 minutes of training.

Once the optimal architecture is selected based on the validation set, another grid-search is conducted for the regularization rate [0.00,0.10,0.25,0.50] and learning rate [0.001;0.01;0.1].

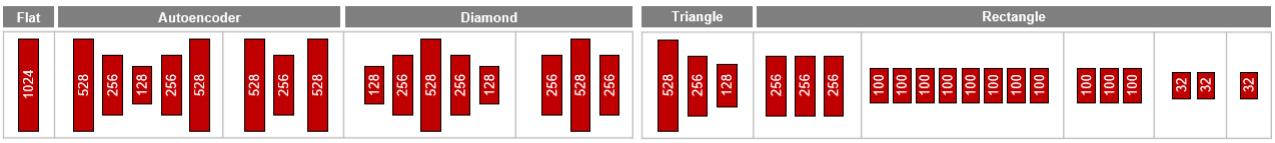


*Figure 3: NN Architectures: a variety of architectures are considered varying both shape and depth*

Determining the optimal architecture for a given problem is an open research question with different architectures performing well in different cases [8]. Model performance has relatively high variance across architectures with larger networks taking longer to train.

Overfitting is major problem with MLPs and therefore regularization may be helpful in ensuring the model generalizes well to unseen data [21]. It appears that 10% regularization offers the best performance on the validation set. Setting the learning rate for the optimizer determines how quickly the model adjusts its weights and is a very important factor in convergence. The learning rate with the highest performance on the validation set is 0.1 (Figure 4).

**Architecture Validation Set Performance (20,000 Samples)**

| # | Model Architecture | Feature Extractor Raw | Surf | CNN | Avg Time | Avg |
|---|---|---|---|---|---|---|
| 1 | [1024] | 10.1% | 18.3% | 80.0% | 114.0 | 36.1% |
| 2 | [100,100,100,100,100,100,100,100] | 10.5% | 15.8% | 79.4% | 33.6 | 35.2% |
| 3 | [528,256,128] | 10.1% | **22.5%** | 80.8% | 70.6 | 37.8% |
| 4 | [256,256,256] | 9.5% | 10.6% | 80.8% | 37.1 | 33.6% |
| 5 | [128,256,528,526,128] | 10.0% | 21.0% | 80.8% | 65.8 | 37.3% |
| 6 | [256,528,256] | 9.9% | 8.8% | 79.8% | 38.1 | 32.8% |
| 7 | [528,256,128,256,528] | 10.2% | 19.8% | 76.9% | 84.2 | 35.6% |
| 8 | [528,256,528] | 10.3% | 20.5% | 81.3% | 113.9 | 37.3% |
| 9 | [100,100,100] | 10.2% | 10.2% | 81.6% | 15.7 | 34.0% |
| 10 | [32] | 10.2% | 16.9% | **83.3%** | 6.9 | 36.8% |
| 11 | [32, 32] | **10.6%** | 15.9% | 80.5% | 6.8 | 35.7% |
| | Average | 10.2% | 16.4% | 80.5% | 53.3 | |

**Hyperparameter Validation Set Performance (20,000 Samples)**

| # | Reg. Rate | Learning Rate | Feature Extractor Raw | Surf | CNN | Avg Time |
|---|---|---|---|---|---|---|
| 1 | 0% | 0.001 | 10.1% | 21.7% | 83.4% | 17.8 |
| 2 | 0% | 0.01 | 9.7% | 20.6% | 83.0% | 13.6 |
| 3 | 0% | 0.1 | 9.9% | 9.1% | 82.7% | 6.6 |
| 4 | 10% | 0.001 | 9.9% | 20.5% | 82.7% | 17.8 |
| 5 | 10% | 0.01 | 10.8% | 20.6% | 83.6% | 14.4 |
| 6 | 10% | 0.1 | **11.1%** | 14.5% | **84.5%** | 13.7 |
| 7 | 25% | 0.001 | 9.6% | 22.4% | 83.4% | 18.0 |
| 8 | 25% | 0.01 | 10.3% | **22.5%** | 83.8% | 15.2 |
| 9 | 25% | 0.1 | 10.0% | 14.6% | 82.6% | 13.8 |
| 10 | 50% | 0.001 | 10.5% | 19.9% | 84.1% | 18.7 |
| 11 | 50% | 0.01 | 9.2% | 11.0% | 83.9% | 7.7 |
| 12 | 50% | 0.1 | 9.7% | 15.0% | 84.6% | 14.1 |
| | Average | | 10.1% | 17.7% | 83.5% | 14.3 |

*Figure 4: Architecture (left) and Hyperparameter Evaluations (right) for NN*

The best performing configuration in terms of speed and accuracy is a neural network with a one hidden-layer architecture (32 nodes) with 10% regularization and learning rate of 0.1. The more significant driver of performance is the feature extractor with performance increasing from raw pixel data to the CNN. This supports the notion that feature engineering and representation are the key to high performance in machine learning [9].

## 3.2 Architecture and Parameters used for the SVM

In this experiment a grid search approach is being evaluated also for SVM. Since this is a multiclass problem, in addition to the main hyper-parameters of a SVM model which are 'kernel' and 'regularization' (*boxConstraint*), a third one is added: the coding design. This coding design evaluates the model between OVO and OVA classification, while the kernel function defines the shape of the decision boundary of the model and the box constraint controls the maximum penalty fixed on margin-violating observations. Box constraint serves as a degree of importance of miss-classification. Increasing the value of the box constraint will decrease the occurrence of wrongly classified images (*hard margin* problem) leading to a higher probability of overfitting, as well as longer training time. Conversely, lower values are computationally faster but might cause underfitting.

Therefore, to derive the optimal *architecture* of a SVM the following values have been assessed:

| Kernel | Box Constraint | Coding Design |
|---|---|---|
| [linear, gaussian, polynomial] | [0.01, 0.1, 0.3, 0.5, 1] | [onevsone, onevsall] |

**Kernel Validation Set Performance (20,000 Samples)**

| # | Kernel | Feature Extractor Raw | Surf | CNN | Avg Time | Avg |
|---|---|---|---|---|---|---|
| 1 | Linear | **10.6%** | **35.5%** | **85.6%** | 1,419.2 | 43.9% |
| 2 | Gaussian | 10.0% | 10.0% | 10.0% | 2,311.1 | 10.0% |
| 3 | Polynomial | 10.0% | 12.1% | 72.4% | 2,981.9 | 31.5% |
| | Average | 10.2% | 19.2% | 56.0% | 2,237.4 | |

**Hyperparameter Validation Set Performance (20,000 Samples)**

| # | Box Constraint | Coding | Feature Extractor Raw | Surf | CNN | Avg Time |
|---|---|---|---|---|---|---|
| 1 | 0.01 | 'onevsone' | 10.8% | **37.2%** | **85.7%** | 190.3 |
| 2 | 0.01 | 'onevsall' | 9.7% | 30.7% | 84.8% | 951.1 |
| 3 | 0.1 | 'onevsone' | **10.5%** | 35.9% | 85.6% | 407.0 |
| 4 | 0.1 | 'onevsall' | 10.1% | 31.2% | 82.9% | 1,641.6 |
| 5 | 0.3 | 'onevsone' | 10.2% | 36.0% | 85.6% | 896.1 |
| 6 | 0.3 | 'onevsall' | 10.1% | 30.8% | 83.0% | 1,839.1 |
| 7 | 0.5 | 'onevsone' | 10.4% | 35.5% | 85.6% | 1,142.5 |
| 8 | 0.5 | 'onevsall' | 10.1% | 29.9% | 83.0% | 1,969.8 |
| 9 | 1 | 'onevsone' | 10.6% | 35.5% | 85.6% | 1,403.2 |
| 10 | 1 | 'onevsall' | 10.3% | 29.9% | 83.0% | 2,246.8 |
| | Average | | 10.3% | 33.3% | 84.5% | 1,268.8 |

*Figure 5: Kernel-Architecture (left) and Hyperparameter Evaluations (right) for SVM*

The best performing configuration in terms of speed and accuracy is a linear kernel with a box constraint of 0.01 and one-vs-one coding (Figure 5). As in the MLP case, performance increases from raw data to Surf and CNN extractor.

# 4 Analysis and Critical Evaluation of Results

Comparing the two models in terms of configuration, SVM has far fewer parameters to select and its properties are far better understood therefore it is easier to grid-search. In contrast, MLP requires more experimentation and domain knowledge in identifying architectures and configurations which are likely to yield strong performance to grid-search.

**MLP Holdout Set Performance**

| Feature Extractor | Train Perfor. | Test Perfor. | Elapsed Time (S) |
|---|---|---|---|
| Raw | 14.3% | 10.4% | 9.0 |
| Surf | 21.8% | 20.7% | 32.3 |
| CNN | 85.3% | **83.5%** | 15.0 |

**SVM Holdout Set Performance**

| Feature Extractor | Train Perfor. | Test Perfor. | Elapsed Time (S) |
|---|---|---|---|
| Raw | 35.2% | 10.1% | 1,090.1 |
| Surf | 49.5% | 37.0% | 164.8 |
| CNN | 99.7% | **85.7%** | 181.0 |

*Figure 6: MLP and SVM Test Set Performance*

SVM with optimally selected hyperparameters outperforms MLP on the holdout test set by a fairly substantial margin (Figure 6). MLP marginally outperformed SVM on the raw data which is consistent with its property of being able to extract information in high dimensional noisy feature spaces [24]. It should be noted that the classes are balanced and therefore a naive classifier guessing randomly would achieve 10.0% accuracy on average.

Contrary to our initial hypothesis, SVM greatly outperformed MLP on the extracted surf features. This turns out to be quite common in image data which can be attributed to the way in which SVM determines decision boundaries [17, 23]. Similarly, SVM outperformed MLP on the CNN extracted features.

Also contrary to our hypothesis, the elapsed time of the SVM was far greater than MLP across feature extraction techniques. This is consistent with other academic research and can be explained by SVMs quadratic run time [17, 23]. MLP implementations are easily parallelized and therefore the networks were trained on a GPU which greatly reduces the training time. The elapsed time for training on the raw data versus the Surf and CNN is substantially higher for SVM which can be attributed to the lack of structure in the raw data.

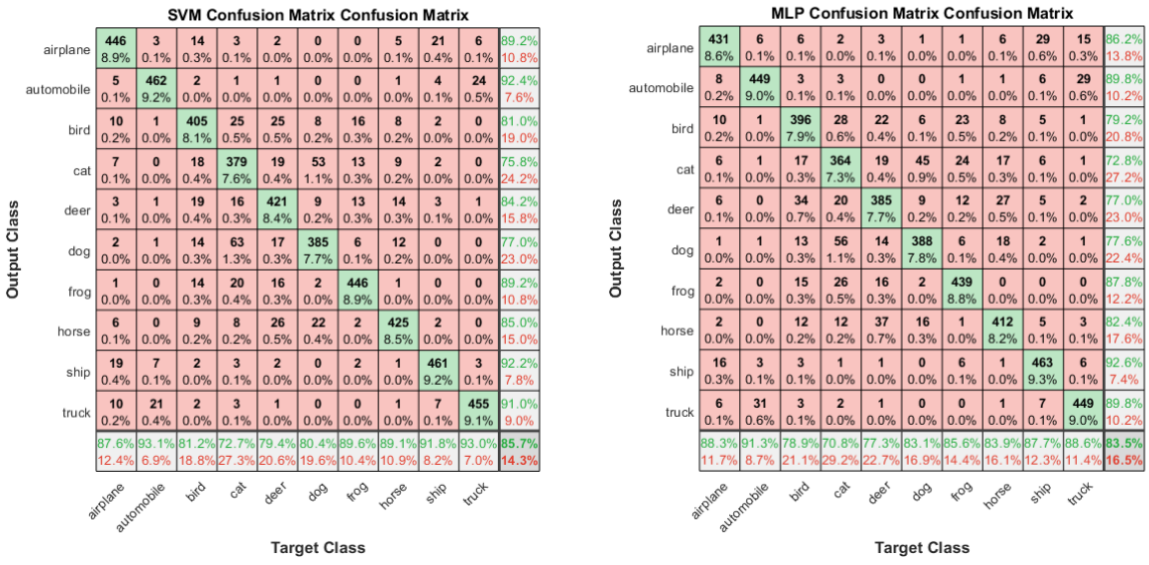


*Figure 7: Confusion Matrix for SVM (left) and MLP (right) on holdout test set*

The most common misclassifications are between cat & dog, truck & automobile, and deer & horse. Both models are similar in terms of classification accuracy and error although interestingly the MLP has a relatively high error rate between deer and birds which results in deer having the second worst performance. The only classes which MLP outperforms SVM are dog and ship which are only marginally better (Figure 7).

Our hypothesis that feature extraction would be a larger factor in driving performance was correct for both algorithms with both models achieving their best performance on the features extracted from the CNN. The accuracy for both algorithms was very close to random guessing using the raw pixel data demonstrating the difficulty in performing classification without meaningful features.

# 5 Conclusion & Future Works

In this study we compared the performances of a Support Vector Machine and a Multilayer Perceptron for a simple multiclass image classification problem. The final results are contrary to our hypothesis. We thought that MLP would achieve higher performance than SVM in terms of accuracy although would take longer to train however the final results were the opposite. Feature extraction still turned out to be orders of magnitude more important the model selection which was consistent with our hypothesis and research.

SVM outperformed MLP using both Surf and CNN feature extractors although MLP marginally outperformed SVM on the raw pixel data making SVM a better model to use for this application assuming no time or computational constraint on training. The classification matrix was a very useful way of visualizing and comparing the final outputs. Moreover, it highlights the difficulties encountered by our classifiers in labeling the correct images such as between dog and cat.

As part of future work, we suggest testing both algorithms using other feature extraction techniques such as HOG or other pre-trained convolutional networks. It could also be interesting to apply some unsupervised learning techniques to represent the data before classification such as using a restricted boltzmann machine, K-means or other unsupervised approaches. One limitation of our analysis is we were restricted computationally and ideally the analysis would be performed on the full dataset with k-fold cross validation to select the best hyperparameters as both models would benefit from an increased amount of training data.

# Bibliography


[1] AI Progress Measurement, *https://www.eff.org/ai/metrics*.

[2] Cifar-10 Dataset, *https://www.cs.toronto.edu/ kriz/cifar.html*

[3] Popular Datasets over time, *https://www.kaggle.com/benhamner/popular-datasets-over-time/code*.

[4] Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., and Arshad, H., 2018. State-of-the-art in artificial neural network applications: A survey. *Heliyon, 4(11)*, e00938.

[5] Awad, M. and Khanna, R., 2015. Support vector machines for classification. In *Efficient Learning Machines* (pp. 39-66). Apress, Berkeley, CA.

[6] Bodapati, J. D., and Veeranjaneyulu, N., 2019. Feature Extraction and Classification Using Deep Convolutional Neural Networks. *Journal of Cyber Security and Mobility, 8(2)*, 261-276.

[7] Coates, A., Ng, A., and Lee, H., 2011, June. An analysis of single-layer networks in unsupervised feature learning. *In Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 215-223).

[8] Deng, L., 2014. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing, 3.*

[9] Domingos, P. M. ,2012. A few useful things to know about machine learning. *Commun. acm, 55(10)*, 78-87.

[10] Garcez, A., 2019. Lecture Notes, *INM427 Neural Computing*.

[11] Guyon, I., and Elisseeff, A., 2003. An introduction to variable and feature selection. *Journal of machine learning research, 3(Mar)*, 1157-1182.

[12] Han, H. and Yang, W., 2014. Object Recognition in Images.

[13] Hinton, G.E., Krizhesvky, A. and Sutskever I., 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in neural information processing systems, 25* pp.1106-1114.

[14] Janocha, K., and Czarnecki, W. M., 2017. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*.

[15] LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K. R. ,2012. Efficient backprop. In Neural networks: Tricks of the trade.*Springer, Berlin, Heidelberg* (pp. 9-48).

[16] Livni, R., Shalev-Shwartz, S., and Shamir, O., 2014. On the computational efficiency of training neural networks. *In Advances in Neural Information Processing Systems* (pp. 855-863).

[17] Meyer, D., Leisch, F., and Hornik, K., 2003. The support vector machine under test. *Neurocomputing, 55(1-2)*, 169-186.

[18] Olden, J. D., and Jackson, D. A., 2002. Illuminating the black box: a randomization approach for understanding variable contributions in artificial neural networks. *Ecological modelling, 154(1-2)*, 135-150.

[19] Pedersen, J. T., 2011. Study group SURF: Feature detection & description. *Department of Computer Science, Aarhus University.*

[20] Smola, A. J., and Schölkopf, B., 2004. A tutorial on support vector regression. *Statistics and computing, 14(3)*, 199-222.

[21] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. ,2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15(1), 1929-1958*.

[22] Wu, T. F., Lin, C. J., and Weng, R. C., 2004. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research, 5(Aug), 975-1005*.

[23] Zanaty, E. A., 2012. Support vector machines (SVMs) versus multilayer perception (MLP) in data classification. *Egyptian Informatics Journal, 13(3)*, 177-183.

[24] Zeki-Suac, M., Pfeifer, S., and arlija, N., 2014. A Comparison of Machine Learning Methods in a High-Dimensional Classification Problem. *Business Systems Research Journal, 5(3)*, 82-96.