Image Classification of Aquatic Macroinvertebrates

ALEX JUSTESEN KARLSEN
Aarhus University - Department of Engineering
au515295@post.au.dk

May 3, 2019

Abstract

This paper documents my work on the in-class Kaggle Challenge; 'AU-ENG-CVML2019'. It is a mandatory submission in the Computer Vision and Machine Learning 2019 course at Aarhus University. The machine learning challenge is to obtain decent classification result on th FIN-Benthic data set consisting of 29 classes of aquatic macroinvertebrates. The paper explains the data set and a way to augment the data set to obtain better results. Furthermore the paper describes the machine learning methods used and the results.

1. INTRODUCTION

Life below water is a critical ecosystem, that experiences a lot of challenging environmental changes. It so important a challenge, that it has found its' way to the UN Sustainable Goals [1]. The Finnish Environment Institute [2] SYKE claims, that the loss of aquatic biodiversity and associated Ecosystem Services is one of the most pressing problems on Earth. SYKE launched the DETECT project [3], which aims to identify the various species of aquatic macroinvertebrates by imagery and classification using computer vision and machine learning.

This paper is a part of the Computer Vision and Machine Learning course at Aarhus University. It documents my work on solving the in-class Kaggle Challenge. The aim of the Kaggle challenge is improving the classification accuracy of classifying the 29 classes of aquatic macroinvertebrates. Machine Learning and state-of-the-art Deep Learning approaches utilizing Convolutional Neural Networks, are used to examine the different methods and get hands-on experience with a real classification challenge.

The rest of the paper is structured as follows; section 2 describes the data set in more depth and a way to augment the data to improve the accuracy. Section 3 describes the machine learning and deep learning methods used in the project. Section 4 describes and discusses the results obtained from applying the methods. Section 5 is a discussion of the overall results and lastly in section 6 the project is concluded upon.

2. DATA SET

The FIN-Benthic [4] data set 2 are used in this paper. The data set consists of 29 different classes of aquatic macroinvertebrates.

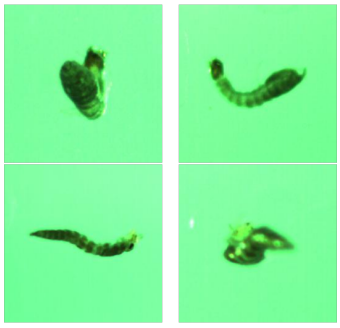

Figure 2.1. *Example of aquatic macroinvertebrates*

The data set counts 11,588 images divided into training set (5830 images), validation set (2298 images) and test set (3460 images). The training and validation set are annotated with class labels, whereas the test set are not. This paper is a part of a Kaggle[1]Challenge with the aim of getting the best classification score on the test set, that are evaluated on Kaggle. For the challenge feature vectors produced by an AlexNet[5] trained on the raw images. The features are the output of the last convolutional layer and flattened into a one-dimensional vector with 4096 features. All images and vectors comes in pairs of images of the same bug taken form perpendicular angles, an attempt to take advantage of the systematic series the images are all images vertically stacked. If we have a pair of x_1 and x_2, then x_1 and x_2 are stacked as a combination of the two;

$$x_1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, x_2 = \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}$$

The vectors now contains $2 \times 4096 = 8192$ features. The concatenated version of the data set are referred to as FIN-Benthic-Concatenated. Futhermore the training and validation sets are merged to form a combined set, to enable Cross-Validation for the different classification algorithms tested in this paper. Cross-Validation and Machine Learning methods used are described in the next section.

3. Methods

Several experiments on this supervised learning challenge have been conducted using a variety of different techniques for classification. The experiments are performed on the encoded data sets; *FIN-Benthic* and an augmented version *FIN-Benthic-Augmented* using the pair-wise information in the original data. The classification methods are; k-Nearest Neighbors (kNN), Linear Discriminant Analysis (LDA), Support Vector Machines (SVM), Perceptron and Neural Networks (NN). Commonly for all these supervised learning algorithm is the need for training. Training an algorithm is an optimization process of a loss criterion. The loss function explains the goodness of the model. Cross Validation (CV) are used for model selection to evaluate the best performing model. All the methods have been applied to the data sets on full dimensions and additionally with dimensionality reduction as a preprocessing step in the classification pipeline. Grid search have been used to search for good hyperparameters. Hyperparameters are user specified parameters that controls the algorithm's behavior [6]. Hyperparameter search have been applied for all classifiers. Furthermore the degree of dimensionality reduction have been investigated in the same manner. Lastly existing Convolutional Neural Network architectures have been used on the raw images using the promise of transfer learning to use a pre-trained model to gain faster and better performing models on small data sets. For the transfer learning task the existing train, validation and test split have been used.

k-Nearest Neighbors (kNN) K-Nearest Neighbor is a distance-based classifier and as the name implies, classifies to the most common occurring class of the k-nearest neighbors [7]. There exist a variety of distance metrics, some example are; Euclidean, Minikowski and Manhattan distance [8].

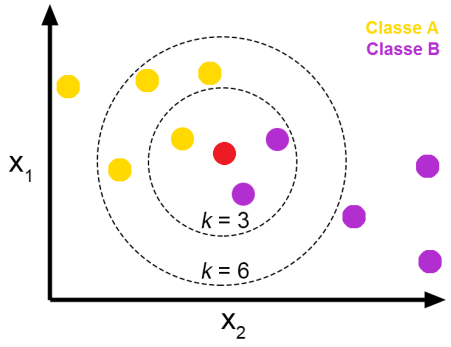

Figure 3.1. *k-Nearest Neighbor classifies the red sample to class B for $k = 3$, however for $k = 6$ it classifies to class A. Credits: Italo José*

[1]Kaggle is an online community of data scientists and machine learners. Red more: https://www.kaggle.com/getting-started/44916

Linear Discrmininat Analysis (LDA) Linear Discriminant Analysis is a linear transformation of the data. The linear transformation is an optimization, that tries to maximize the distance between classes and minimize the distance within a class[8]. Figure 3.2 shows how LDA discriminates two classes.

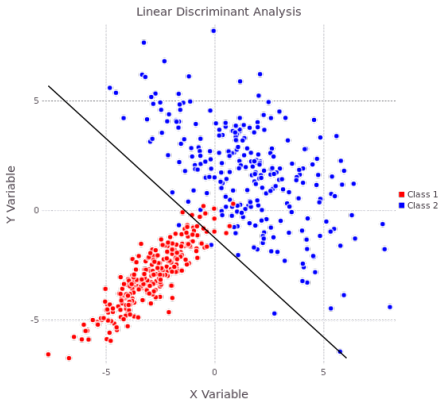

Figure 3.2. *LDA creates a linear decision function. The linear decision function is a decision hyperplane which discriminates classes. Credits: Tim Thatcher*

LDA can also be used for dimensionality reduction, this would be further explained later.

Support Vector Machine (SVM) SVM can be used for classification. SVM tries to find the decision hyperplanes, that maximizes th distance between classes [9]. The decision hyperplane are given by a linear function $w^T x + b$, when the function is positive it predicts the class to be present, and intuitively the class is not present, when the function is negative [6]. The Support Vector Machine uses support vectors to define the decision hyperplane. The support vectors are vectors of the training data, that maximizes the margin between to classes. The original SVM proposed by Boser et al. [10] utilized a hard-margin. The hard-margin is the optimal decision hyperplane, however it only works well when the data is linearly separable. Vapnik et al. [11] proposed a soft-margin, which is more robust to noise by introducing some slack i the fom of the hinge loss function.

One of the most promising contribution associated with SVMs is the kernel trick. The kernel trick makes not only SVM useful when data is not linearly separable, but all machine learning algorithms, that can be rewritten exclusively in terms of dot product between examples. The kernel trick is a mapping of the data into a higher dimensional space - even with indefinite dimensions. The kernel trick is equivalent to transforming all data points into higher dimensions, but with a lot less computational demands. In the high dimensional space a linear decision boundary can be obtained. One of the most widely used kernels are the gaussian kernel or radial basis function (RBF), which maps to an indefinite dimensional space [6].

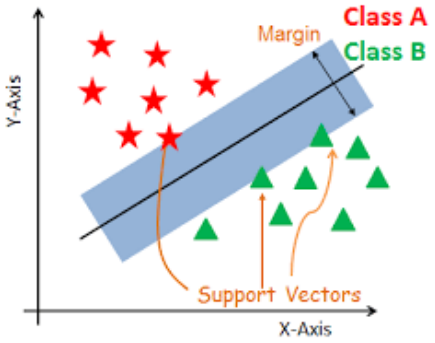

Figure 3.3. *A binary SVM constructing the decision hyperplane, that maximizes the margin between the support vectors of the two classes. Credits: Avinash Navlani*

Perceptron Perceptrons is a single layer Neural Network. The perceptron was set forth by Frank Rosenblatt [12], as a way of mimicking human perception. Perceptrons produces a linear decision function and only performs well on linearly separable data.

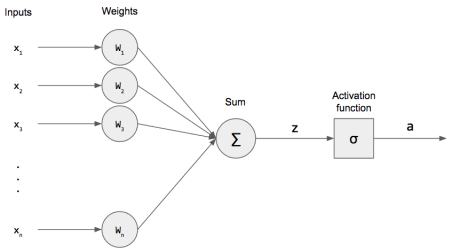

Figure 3.4. *The perceptron takes a feature vector as input. All the feature are multiplied by a weight. The weight are then summed to what is called a weighted sum. An activation function are applied to the weighted sum. The output of the activation function is the classification. Credits: Stanley Obumneme Dukor*

The perceptron are trained by iteratively updating the weights to produce the correct output for the training data. The loss function are rather simple, if the sample is correctly classified th loss function will output zero and it will output 1 if misclassified. This means the perceptron's weight are only updated when is fails to classify a sample.

Neural Network (NN) Neural Networks extends the perceptron. This ideas was sparked by progress within biology, that mapped human perceptions into a network of neurons i.e. perceptrons. The first working Artificial Neural Network (ANN or NN) was completed by A. G. Ivakhnenko and V. G. Lapa [13]. The perceptrons where extended by adding more layers between input and output referred to as hidden layers. The NN is capable of producing non-linear decision functions, thus overcomes the short comings of the perceptron.

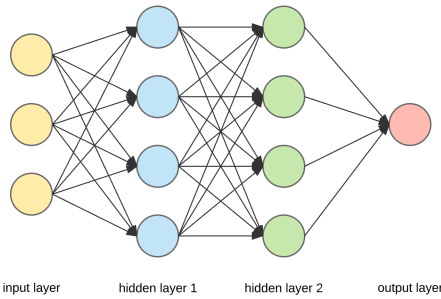

Figure 3.5. *Neural Network with two hidden layers. Each layer is formed by a number of neurons. A fully-connected dense neural network receives all output from previous layer as input. The input are multiplied by the neuron's weight and a bias is added. At last the non-linear activation function is applied. Credits: Arden Dertat*

Training a NN is way more complex than a perceptron. The training are done using feed-forward and back-propagation. Training vectors are given to the network, where all nodes do their operations upon reception. The output from this layers are then fed forward to the next layer. When it reaches the end of the network an error term is computed using the loss function. The error are then back-propagated throughout the network, where the nodes' contribution to the error is computed. Using gradients the network's weights are optimized.

Convolutional Neural Network (CNN) CNNs are today the golden standard to obtain good classification results on images. AlexNet was the first CNN, that made a breaking result on the ImageNet Challenge in 2012. Since then a lot of then architectures have been proposed and introducing smaller changes to improve just a little is published every now and then. Traditional neural network does not scale to task as images increases in size. CNNs' does not take single pixels as input, but instead convolutional layers are added to the architecture to inspect local regions of pixels to accommodate spatial correlation in images also known as filters. The network then learns filters, that represents features such as a diagonal edge, a blob, etc. Throughout the network the image is down sampled by pooling layers and

eventually classified using typically a softmax layer [14].

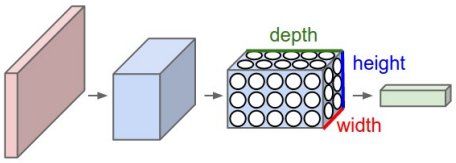

Figure 3.6. *A CNN arranges its neurons in three dimensions. The layers of the CNN transforms the 3D input to a 3D output of neuron activations. The input layer holds the image and the width and height equals the dimensions of the image, and the depth equals 3 i.e. the color channels 'RGB' of the image. Credits: Andrej Karpathy*

GoogleNetV4 aka. InceptionResNetV2 [15] is a very deep CNN, in fact 572 layers deep. It is one of the top-scoring networks on the ImageNet Challenge with a top-5 accuracy equalling 0.95. [16] It is available trough Keras with pretrained ImageNet weights.

This sums of all the classification methods used in this paper. Next I describe practices concerning machine learning.

Dimensionality Reduction Dimensionality reduction as the name implies down scales the feature space into a lower dimensional space. The techniques used in this paper is the already covered LDA and Principal Component Analysis (PCA).

Principal Component Analysis (PCA) PCA is an unsupervised learning technique. It learns an orthogonal linear transformation onto a smaller feature space. The number of dimensions i.e. principal components are a hyperparameter. PCA tries to preserve as much as the information i.e. variance as possible. The first component i.e. dimension is a projection a long the axis of the data with the most variance, the second component is orthogonal to the first component and describes the axis with the second most variance. The third component then describes the axis with third most variance and so on. The figure 3.7 the component axis as vector called $PC1$ and $PC2$.

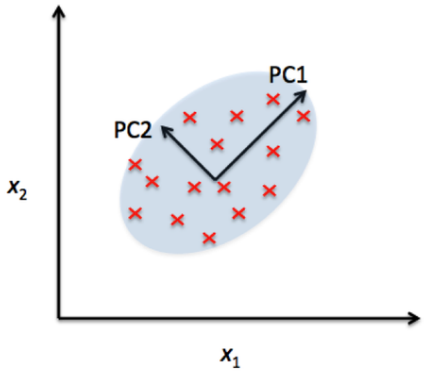

Figure 3.7. *Principal Component Analysis in a 2 dimensional space. PCA rotates the data to be perpendicular to the principal components axises. Credits: Sebastian Raschka*

Linear Discriminant Analysis (LDA) As mention LDA can be used for dimensionality reduction as well. One must specify the n number of components i.e. dimensions to transform the data onto, if the n are less than the original data dimensions D, then the number of dimensions will be reduced.

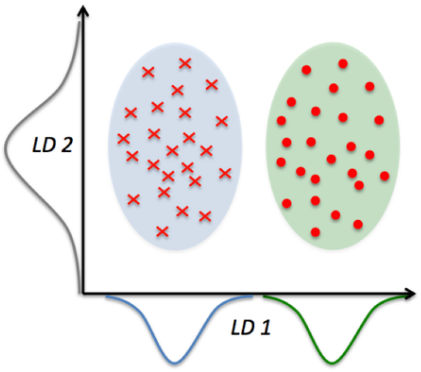

Figure 3.8. *Linear Discriminant Analysis project the data onto and axis, that maximize the distance between the classes and minimizes the distance within the respective classes. LD1 is a good project, whereas LD1 is not. Credits: Sebastian Raschka*

Model Selection Model selection covers procedures to train the best model and avoid overfitting the model to the training data. Overfitting is the pitfall of having a model, that describes

the training data too well, thus suffers not to generalize the true underlying structure of the data. Although other methods exists the two used in this paper are Validation Holdout and Cross Validation.

Validation Holdout Validation Holdout takes a fraction of your available labelled data and hold it out of the training procedure.

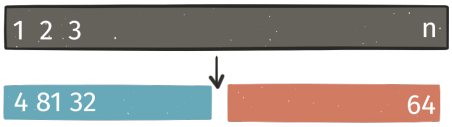

Figure 3.9. *The validation subset is then used to validate the model's performance on data it has not been training.*

The assumption is that the holdout set resembles the true data, since it has been drawn from the same distribution [6]. If the model performs well on both training and validation it is assumed to be a good model. The split is typically made $70/30$ or $80/20$ for training data and validation data respectively.

Cross Validation (CV) Cross Validation extends on the holdout validation. k-fold Cross Validation create k experiments compared to one. The entire data set are split in k folds, for every k step one subset is for validation and the remaining $k-1$ fold are used for training [6]. Figure 3.10 illustrates the idea.

Figure 3.10. *Cross Validation ensures that the model have been trained on all the data and validated as well. The selection folds are typically either $k = 5$ or $k = 10$.*

Hyperparameter Search There are mainly two approaches to search for hyperparameters; Random Search and Grid Search other than heuristic guessing. In this paper Grid Search are used.

Grid Search Grid Search is an approach to seek better hyperparameters for your model. You specify a set of options for all hyperparameters and then search all possible combinations in the grid.

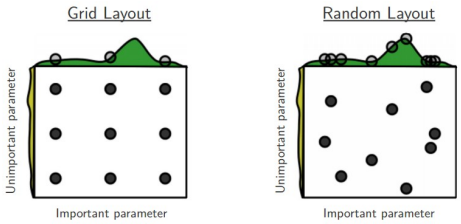

Figure 3.11. *Grid search is a systematic approach to explore a broader space of hyperparameters although at the cost of additional time. The down-side of this approach is, that it is still based on intuition of good hyperparameters. The random approach are also widely used and might be preferred, as it may find a sweet spot, that goes beyond human intuition, however that is no guarantee, and it is still time costly [17]. Credits: Bergstra and Bengio*

4. Results

Two main experiments have been conducted; the first experiment involves applying the aforementioned machine learning methods on the feature extracted vector. The second experiment involves applying deep learning practices on the raw image data.

Experiment 1: Machine Learning

The machine learning experiments are implemented in Python 3 using the SciKit Learn [18], except the Neural Network that are implemented in Keras [19] using Tensorflow [20] backend.

All classifiers have been run on the FIN-Benthic data set and on the FIN-Benthic con-

catenated data set using Grid Search and CV to find the best hyperparameters, except the neural networks. As grid search has been used to find the best set of parameters for each of the classifiers and on the different data sets, it has not given the same results for best hyperparameter settings. Only the best performing setting of hyperparameters for each classifier are presented in this paper.

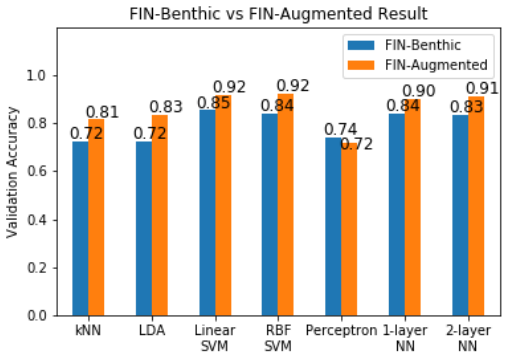

Figure 4.0. *The best setting for each of the classifiers in terms of accuracy. Generally the classifiers perform best on the concatenated data set. The assumption about concatenating the pair sample of the same bug holds true.*

I added PCA to the classification pipeline and ran the same grid search on all the classifiers on both data sets. For all runs the dimensionality have been reduced to $n = \{128, 256, 512, 1024\}$ principal components. PCA add additional time to the pipeline, however the time to train and predict on fewer dimensions significantly improved. Although the dimensionality have been reduced with a large margin it does not have huge impact on the classification accuracy for all classifiers. The ones that showed the largest improvements was kNN and LDA.

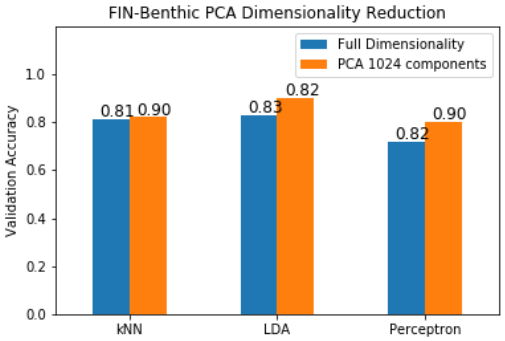

Figure 4.0. *Adding PCA 1024 components to the pipeline improves classification accuracy by 5 point for kNN and 11 points for LDA. The other classifier showed similar results to full dimensions. The two improved classifiers seems to more prone to noise, thus PCA shows these big improvements. However PCA might still be useful for the other classifiers as it reduces training time by a large margin.*

I did the same procedure for the augmented data set.

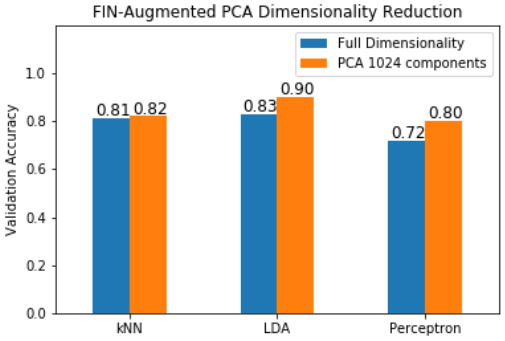

Figure 4.0. *Adding PCA 1024 components to the pipeline improves classification accuracy by 1 point for kNN, 7 points for LDA and 8 points for the Perceptron. The other classifier showed similar results to full dimensions. It seems the perceptron is better as discriminating the reduced version of the augmented data set. The PCA transformation seems to give more linearly separable features.*

Experiment 2: Deep Learning

I set up an InceptionResNetV2 and initialized it with pre-trained ImageNet weights. The input

layer i changed to equal the image size, and I added a softmax classifier equivalent to the number of classes i.e. 29. I used data augmentation to generate artificial samples and counter avoid overfitting. I trained the network for 100 epochs, but was not able to get validation accuracy above 0.80. Unfortunately due to limited access to GPUs, I could not do several runs to perfection the hyperparameter and augmentation to obtain higher accuracy.

5. Discussion

Taking advantage of the pairwise structure of the FIN-Benthic data set significantly improves the machine learning algorithm's classification performance. The double amount of information seems very useful and can be backed up by looking at the raw images. In most cases the images resembles each other, however there are cases, where the images seems to be taken from an unfortunate angle. The doubled information seems to handle this problem. Nonetheless inspecting the predicted class labels, one can see, that they are still not completely pairwise segmented. A postprocessing step if using a probabilistic classification model can be proposed. Summing the outcome of class probabilities for the pair and classify to the most probable for the pair could be advantageous.

A future step is to look at random search instead grid search. Although improvements where found using grid search and I started with coarse grids and narrowed down the search space using finer grid, there might be combinations, that I did not consider and which could have given better results.

I have tested out a set of different classification method and the best performing seems to be Support Vector Machines and Fully-Connected Neural Network for the vectorized AlexNet features. It would have been interesting to train a state-of-the-art CNN over a long period of time, to see if the features from such a network would in fact improve the features, thus gives a better accuracy for all the methods. Nowadays architectures like GoogleNetV4 accuracy of 0.803 on the ImageNet data set[15], which have the size required to train such deep models. Transfer learning reuses the weights from e.g. ImageNet to solve a another classification task. The model can be fine-tuned to better fit the special cases of the new data set.

6. Conclusion

I have gained a lot of experience with a practical machine learning challenge. I used the pair-wise information in the data set to optimize model accuracy to a 0.92 validation accuracy and ≈ 0.86 test accuracy on the Kaggle Challenge's test set. I struggled with deep learning performance and did not have the resources to overcome the challenges.

References

[1] "Transforming our world : the 2030 agenda for sustainable development," *UN General Assembly*, accessed: 8 April 2019.

[2] K. Meissner, "Detect." [Online]. Available: https://www.syke.fi/en-US/Research__Development/Research_and_development_projects/Projects/DETECT

[3] E. Riabchenko, K. Meissner, I. Ahmad, A. Iosifidis, V. Tirronen, M. Gabbouj, and S. Kiranyaz, "Learned vs. engineered features for fine-grained classification of aquatic macroinvertebrates," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, Dec 2016, pp. 2276–2281.

[4] K. Meissner, "Fin-benthic." [Online]. Available: https://etsin.avointiede.fi/dataset/urn-nbn-fi-csc-kata20170615175247247938

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. USA: Curran Associates Inc., 2012, pp. 1097–1105. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999134.2999257

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[7] A. Karpathy, "Image classification." [Online]. Available: http://cs231n.github.io/classification/#knn

[8] A. Iosifidis, "Course notes: Introduction to machine learning," Blackboard, 2017.

[9] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[10] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. ACM Press, 1992, pp. 144–152.

[11] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995. [Online]. Available: https://doi.org/10.1023/A:1022627411411

[12] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, pp. 65–386, 1958.

[13] A. G. Ivakhnenko and V. G. Lapa, "Cybernetic predicting devices," p. 250, 04 1966.

[14] A. Karpathy, "Convolutional networks." [Online]. Available: http://cs231n.github.io/convolutional-networks/

[15] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016. [Online]. Available: http://arxiv.org/abs/1602.07261

[16] F. Chollet, "Keras applications." [Online]. Available: https://keras.io/applications/#inceptionresnetv2

[17] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 281–305, Feb. 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2503308.2188395

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[19] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[20] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/