Relevance of classical models in the paradigm of convolution neural networks

Kevin Pan, Pujan Biswas
SUID: 06606125, 06425297
Stanford University, Stanford, CA 94305

Abstract:
In this report, we have analyzed the performance of Convolution neural networks (CNNs) compared to classical models over meteorological datasets. We have utilized a pre-processing technique that has been described to extract the features and utilized same datasets for the training and comparison. The report concludes by giving key insights received from the project including feature selection. We also believe that CNN performance is guided by the data collection process.

Key words: Computer vision, General machine learning, Models, Neural networks

Introduction:

This project aims to explore the limitations of the classical models and CNN and compare their image classification performance on datasets that are well defined. Datasets used in this work were satellite cloud images obtained from NASA-IMPACT-AWS server, as we believe meteorology is one of the most challenging vision problems. Additionally, a few classes namely cloud street, traverse cirrus bands etc., have been selected for multiclass classification. To utilize the best prediction capability of CNN, we have used specific datasets that have clear cloud patterns for feature. For the classical models, this project has used SVM, Random Forest, and Logistic Regression to classify based on the extracted features. We conclude with a short comparison between our obtained results.

Motivation:

CNNs have picked up pace exponentially in the recent years. Even during the current COVID-19 pandemic, CNNs started to show X-ray image diagnostic capabilities. This was particularly highlighted in an Optimized Convolutional Neural network was proposed by Goel et. al. [1] for the automatic diagnosis of COVID-19 optimizing all the hyperparameters. With AlexNet being equipped with transfer learning, classical machine learning models started to fade out of prominence. Later, as VGG-16 came in, CNN took higher precedence over the known classical models such as SVM and Random Forest in terms of complex image classification. Similarly, with the emergence of ImageNet and availability of large scale open-source datasets, CNN quickly rose to the top and became the state-of-the-art for data categorization in computer vision applications. However, training a deep CNN from scratch is computationally expensive and required very large datasets for training commencement [2]. Rajpurkar et. al. also analyzed whether ImageNet architectures are unnecessarily large for training. They also provided the caveat that many deep learning methods including ImageNet typically rely on pretrained models for the development which can create unnecessary overhead [3]. The pressing question of the complexities of CNN blackbox compared with simpler models should be deeply understood.

Dataset and Features:

The dataset consists of 1900 positively classified colored images and 4901 negatively classified images. We have cropped each image to a square image and scaled to 255×255×3 (width × height ×channel) for processing. Thereon, these images were fed into the CNN as image features. In order to extract better features for the classical models, the images were processed further to separate the cloud pattern from the landscape/background. The 3D RGB colored image was mapped into 1D grayscale image using the Eqn. 1 and passed on through ReLu using Eqn. 2

$$X_{Gray} = 0.2989\, X_R + 0.5870\, X_G + 0.1140\, X_B \tag{1}$$

$$x_{i,j} = max\left(80, x_{i,j}\right) \;\forall\; x_{i,j}\; X_{Gray} \tag{2}$$

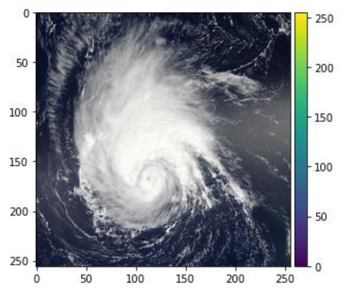
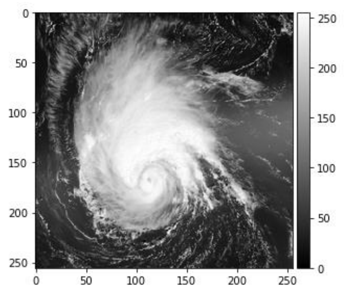
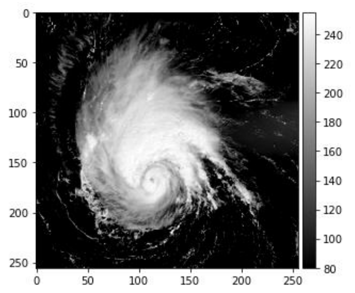
Cropping and Scaling	**Grayscale Conversion**	**ReLu Function**
		

Figure 1: Dataset analysis

Six features were then extracted according to the following steps:

1. Average brightness was calculated summing up all pixel brightness
$$avg\_brightness = \frac{1}{w * h}\sum_i^w \sum_j^h x_{i,j}$$
2. Centroid location is the Euclidean distance from the image centroid to the image center.
$$centroid\_loc = \sqrt{(\mathrm{w}//2 - \mathrm{cx})\text{^}\ 2 + (\mathrm{h}//2 - \mathrm{cy})\text{^}2}\; where\ (cx, cy)\ is\ the\ centroid$$
3. Symmetry is obtained by the average pixel overlap in the original image and 90-degree rotation
$$symmetry = count\left(\sum_i^w \sum_j^h \left(x_{i,j} - x_{i,j}^T\right) > 10\right)$$
4. Centroid density is the average pixel brightness for pixels closer to the centroid.
$$centroid\_density = \sum_i^w \sum_j^h x_{i,j} \in x_{cx\pm\ 0}, x_{cy\pm 10}$$
5. The vertical_val and the horizontal_val is the average brightness on the vertical edge plot and the horizontal edge plot.

vertical_val = avg_brightness (vertical_edge_plot)

The PCA plot is generated with the features scaled down to 2. The red ‘o’ represents the “yes” labeled data and the blue ‘x’ represents the “no” labeled data.

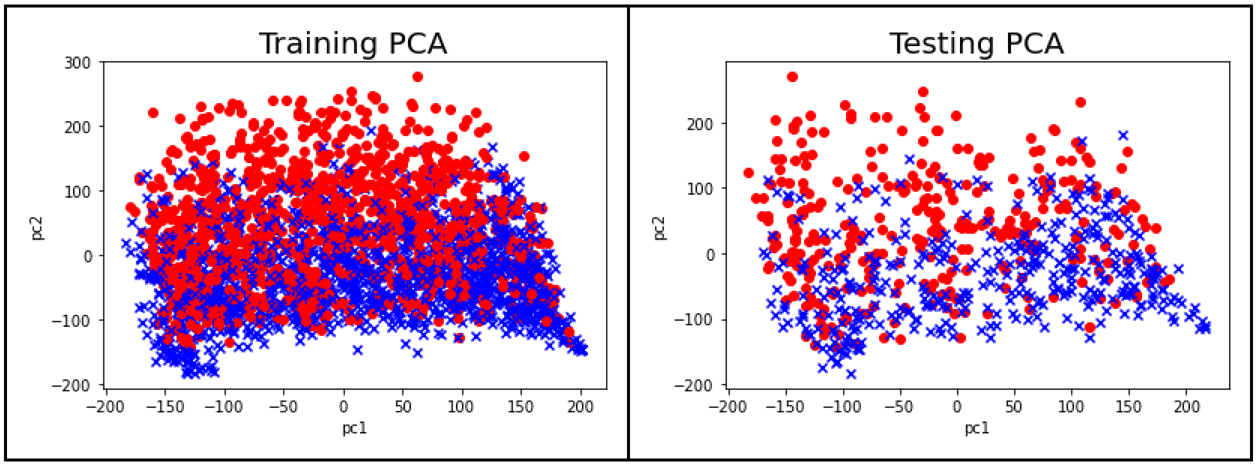


Figure 2: PCA plot for training and testing data

Methods:

1. **SVM**, which maximizes margin between the separating hyperplane using $w^T X + b = 0$ and the nearest elements (support vectors) is the first model. The distance, defined using Eqn.3 leads to objective function maximization using Eqn.4. Eqn.5 defines the RBF kernel used in this work.

$$dist\left(x^{(i)}, boundary\right) = y^{(i)}\left(w^T x^{(i)} + b\right)/||w||_2 \quad (3)$$

$$minimizing\ 0.5||w||_2^2\ such\ that\ y^{(i)}\left(w^T x^{(i)} + b\right) > 1 \quad (4)$$

$$k(x, x') = exp\left(\frac{||x - x'||_2^2}{2\sigma^2}\right) \quad (5)$$

2. **Random Forest**, which is an ensemble of many decision trees by repeatedly resampling the training data has been used in the current work. A single decision tree works by splitting the input data into 2 subcategories at each branch, therefore very prone to overfitting. However, limited depth combined with bootstrap aggregation such as random forest helps overfit reduction. Gini impurity helped measure element frequency being incorrectly labeled based on the distribution.

3. **Logistic Regression** mapped the input linear combination $(\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \theta^T X)$ into [0,1] domain using the sigmoid function $(h_\theta(x))$. The loss function defined using Eqn. 6 has been minimized using by Eqn. 7 via gradient descent delivering the weights as output.

$$L(\theta) = \sum_{i=1}^{N} y^{(i)} log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(h_\theta\left(x^{(i)}\right)\right) \quad (6)$$

$$\theta_j \coloneqq \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right) x_j^{(i)} \quad (7)$$

4. The **CNN model** performed forward and backward propagation until convergence. During the forward propagation step, an input image (width * height * channel) passed through many convolutions and pooling layers and was converted into a flattened 1D array. The array then passed through a few multi-layer perceptron layers which provided an output of prediction that matches the dimensions of the ground truth. The convolution layer used many filters (weights) to extract different feature maps. During the backpropagation, the gradients for each weight matrix are calculated using the chain rule from the output gradient to the gradient of w_1. Finally, the weights for each convolution and fully connected layer are

calculated using $w_i \coloneqq w_i - \alpha \nabla w_i$. The loss function used in this work is Binary Cross entropy loss defined by Eqn.8

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i * \log\big(p(y_i)\big) + (1 - y_i) * \log\big(1 - p(y_i)\big) \tag{8}$$

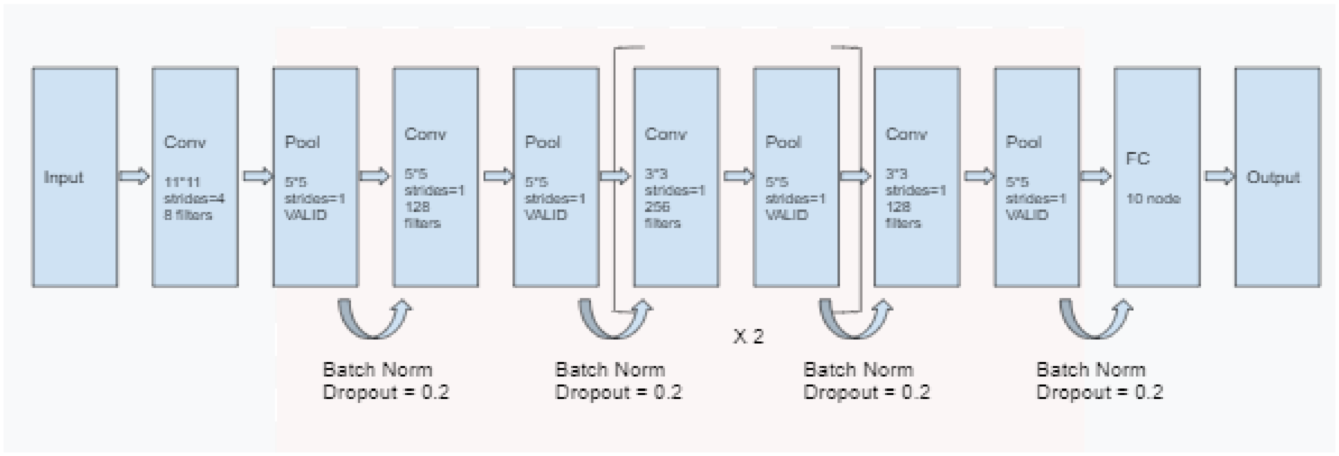


Figure 3: Model flowchart

Overall, the model design is as shown in Figure 3. Batch normalization was added to reduce the vanishing gradient problem and speed up the training. Dropout layers were added after the batch normalization to introduce more noise and reduce overfitting. In each convolution layer, the ReLu activation function was used along with L_2 regularization. The sigmoid activation function was used at the output layer. Finally, early stopping was used to further reduce overfitting.

Visual results:

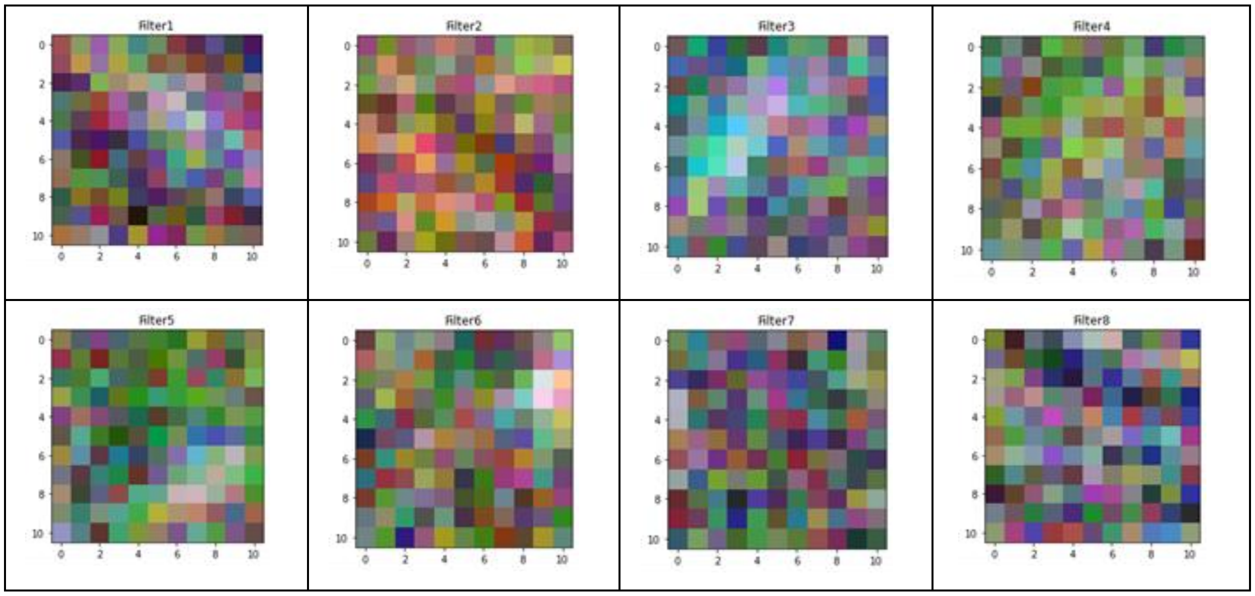


Figure 4: **CNN Filter Visualization. The filter weighs the first convolution layer.**

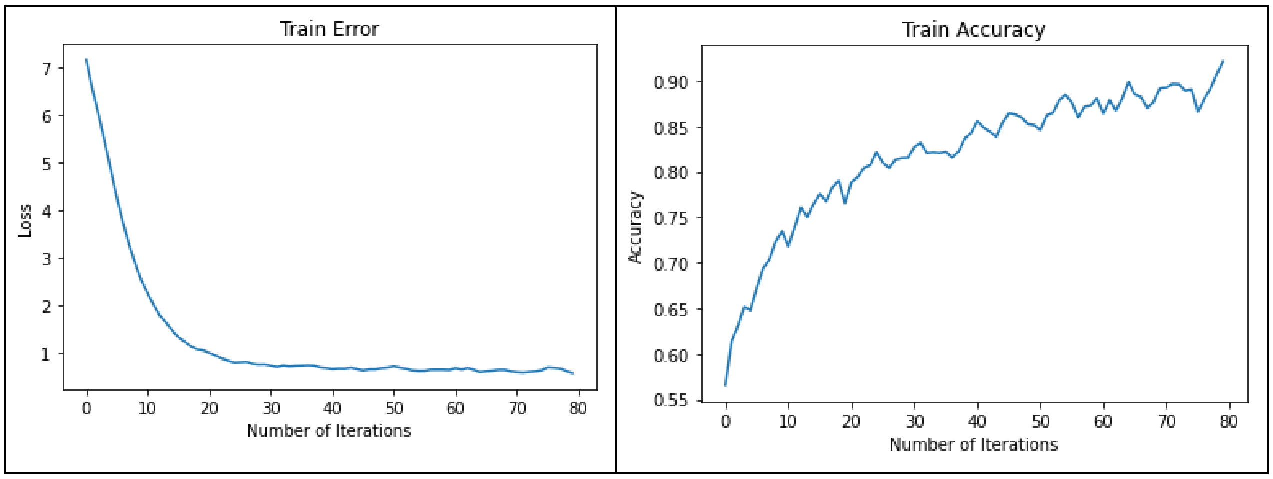


Figure 5: CNN Training Curves

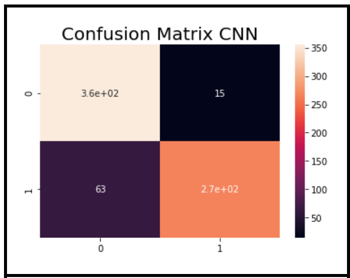
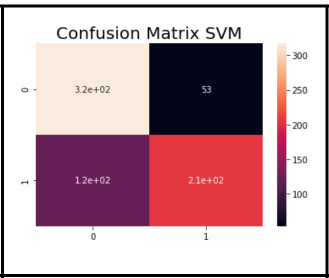
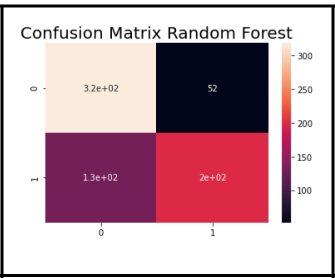
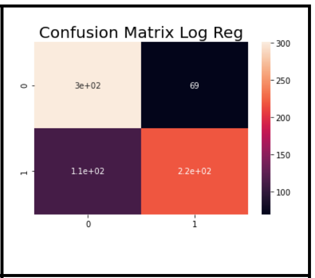
			
Testing accuracy: 88%	Testing accuracy: 75%	Testing accuracy: 74%	Testing accuracy: 74%

Figure 6: Confusion Matrix Plot and Accuracies

Discussion and Conclusion:

In this work, it has been seen that the CNN model performed according to general expectations on the chosen dataset with decreasing training loss increasing model accuracy. From the Figure 4, close resemblance to white patterns of clouds can be observed. Since the Adam optimizer has been used, the curves mimic the stochastic nature of SGD. On the contrary, the classical models had poor performances with the training accuracy being close to the testing accuracy. This was expected since the two classes have too many overlaps, and those models were not complex enough to have severely overfitted. With Figure 6, it can be identified that the CNN model outperforms the classical models significantly.

We believe that these results can be attributed to data collection. When examining the raw image, it was clear that the images were collected under different settings, zoom, resolution, weather, etc. Some images captured a corner of the cirrus clouds and others captured them entirely. An additional challenge is collecting the localized features. Unlike facial recognition, in some satellite images, it is difficult to pinpoint what exact features to look for due to the large number of contributing factors. Meanwhile, the CNN model takes the images as input data and automatically extracts features/filters.

To conclude, when classifying image data, even if the features are well defined, the data needed to be collected under a controlled environment in order for the machine feature extraction to be effective. The model can only perform as well as the inputs (features) fed into the model. The classical models especially demand good and consistent features such as handwritten digit images or reinforcement learning image data in a controlled environment. Another solution is to do extensive data cleaning or to have humans measure and record image features by hand, both of which can be very costly. However, it is possible for the classical models to outperform the CNN models given the perfect features. To answer the final question: Is CNN better than the classical models on image data? The answer, well, is “It mostly depends on the environments in which the data is collected and the qualities of the features can be extracted”.

Reference:

[1] Goel, T., Murugan, R., Mirjalili, S. et al. OptCoNet: an optimized convolutional neural network for an automatic diagnosis of COVID-19. Appl Intell 51, 1351–1366 (2021). https://doi.org/10.1007/s10489-020-01904-z
[2] A. Garg and N. Karimian, "Leveraging Deep CNN and Transfer Learning for Side-Channel Attack," 2021 22nd International Symposium on Quality Electronic Design (ISQED), 2021, pp. 91-96, doi: 10.1109/ISQED51717.2021.9424305.
[3] 2021A Ke, W Ellsworth, O Banerjee, AY Ng, P Rajpurkar10.1145/3450439.3451867Proceedings of the Conference on Health, Inference, and Learning
[4] https://github.com/NASA-IMPACT/data_share
[5] https://en.wikipedia.org/wiki/Decision_tree_learning

Contributions:
Kevin Pan: Model Building, Report.
Pujan Biswas: Data Cleaning, Feature extraction, Report.

Libraries:
tensorflow, keras, sklearn (svm, RandomForestClassifier, LogisticRegression, PCA, confusion_matrix), skimage (prewitt_h, prewitt_v), numpy, matplotlib, csv, pandas, seaborn, PIL, os