4511W - Minnesota Plant Classification using CNN

chanx411, romxx014

April 2020

Abstract

In this project, we have decided to attempt to classify several Minnesota local wildflowers. Using a convolutional neural network (CNN) developed using Python with TensorFlow, we evaluated the efficiency of several architectures and activation functions to determine what configuration of a CNN would be best fit to classify wildflower images. Networks were evaluated on total accuracy before overfitting began, and we determined a CNN using Rectified Linear Unit and Softmax as activation functions was the best option to classify these wildflowers. The neural network was able to achieve a success rate 2.5x better than random when classifying the flowers. In addition we discuss neural networks connection to AI as a whole, real-world examples and other existing solutions for image classification.

1 Introduction

Classification problems are one of the simpler use cases for a Deep Neural Network. They essentially look at an image or an array of data points and give a guess about whether or not that belongs to a given category based on many training samples. For example, a classifier could look at a lung scan and give a probability of being infected by the novel coronavirus based on scans it has seen before.

Despite there are 18k types of plants in the United States, we cannot personally name even a small fraction of those plants across the country. Our goal is to use a neural network to classify a portion of those plants based on species. This would be a classification problem in which the neural network would determine a probability/likelihood that an image is a member of a certain species (class). In order to best classify these local plants, we plan on using a deep neural network with multiple techniques to improve the speed and quality of training, including convolution, image augmentation and data normalization.

We would like to try to classify a set of flowers native to Minnesota because we personally know very little about different species of flowers. This could potentially be expanded later on and used in a mobile application to help people of all ages go out and learn more about the ecosystems around them. It could also aid in wildlife safety, as it could possibly help identify dangerous or poisonous plants that one should stay away from.

2 Finding an Intelligent Solution

Performance, Environment, Actuators, and Sensors (PEAS) analysis is a good way to first look at an agent to solve the problem at hand. The performance measure we used for this problem was accuracy. We needed our agent to accurately identify what flower was in the image, and the only way for it to improve was by making itself more accurate. The environment for this problem is the images of flowers, themselves. Using each pixel value, it should be possible to determine the species in the image. The actuators our agent used were functions on the pixel values of the images. These functions used over 30 million parameters to help it become more accurate in its task. The sensors used by the agent to "see" the environment takes the form of an input layer for the neural network.

The environment our agent is working in has many properties: fully observable, single agent, deterministic, episodic, static, continuous, and unknown. Knowing these features can help determine how the agent is set up and works, and this helped us decide to use a neural network.

The environment consisting of images of flowers is a fully observable environment because there is enough information in the image to determine its species. There is only one agent operating in this environment, so there won't be added bias or complexity from other agents. It is deterministic because there is enough information in the images to determine the species that is being depicted. This environment is episodic because an image from before does not affect the classification of the current image. It only depends on what is currently being looked at. This is a static environment because once the agent sees the image, there will not be any changes until it has made its decision. This type of environment makes solving the problem easier because there is less for the agent to account for. Image classification is a continuous environment because each pixel can theoretically have any value. However, this is time discrete because of there being no role in time for this problem.

Neural networks as a whole serve as a good way to find intelligent solutions, and there are many different architectures, purposes and uses for all of these networks. By transforming data into some sort of meaningful conclusion, neural networks have the capability to learn and understand patterns that could not be quickly determined by humans, if at all.

3 Related Work and Existing Solutions

There are several ways to approach this image classification problem with different methods potentially using data augmentation, convolutions, unsupervised learning (k-means), supervised learning, and deep neural networks. The most basic option available is using a deep neural network (DNN) to classify images because they have proven effective at finding higher-level patterns. According to Sladojevic et al. [10], "Neural networks, with their outstanding ability to derive meaning from complex or imperfect data, can be applied for extracting patterns and detecting trends that are too difficult to notice by humans or computer techniques," meaning classification through a neural network is far superior to the human eye or standard search techniques.

Additionally, it is possible to use convolutional layers. Convolutional layers are layers that can be added to a DNN that assign weights and biases to different parts of the images, which allows the network to learn more efficiently overall. This combination of at least one convolutional layer and a DNN form a convolutional neural network (CNN). According to Connor Shorten, et al. [2], "convolutional layers sequentially downsample the spatial resolution of images while expanding the depth of their feature maps. This series of convolutional transformations can create much lower-dimensional and more useful representations of images than what could possibly be hand-crafted." By giving a lower-dimensional yet meaningful image that represents the whole, the network will process and train over the data quicker and give more accurate results based on the convolution.

Another technique that can be utilized to enhance the learning of the network is data augmentation. Data augmentation is altering the image (potentially by skewing it, flipping it, greyscale, etc). Doing this allows the network to learn over a greater sample of images and adds variations to images to decrease the likelihood of overfitting. According to [2], "Data Augmentation approaches overfitting from the root of the problem, the training dataset... In classic discriminative examples... software must overcome issues of viewpoint, lighting, occlusion, background, scale, and more. The task of Data Augmentation is to bake these translational invariances into the dataset such that the resulting models will perform well despite these challenges."

Other than using a CNN, another way to potentially classify images is using Support Vector Machines (SVM). A SVM's goal is to find a N-dimensional plane (where N = number of features to classify) such that all data points fall within specific categories of that plane. A "support vector" is a data point that lies close to the "margin," or the distance that N classifications are from each other. The goal is to maximize this margin, potentially by deleting support vectors, which comes together to create a Support Vector Machine. A SVM can be used either alongside an neural network, like a CNN, or work independently in image classification problems by breaking data points into classes by using a N-dimensional plane. This is demonstrated by Hasan, et al. [4], claiming SVMs can be applied in detection framework "due to its outstanding generalization capability and reputation in the training data set to achieve high accuracy."

Another way to classify images is by using a decision tree. According to Radmila Jankovic [6], who used decision trees to classify cultral images, "[Decision Tree's] main advantage is that there is no assumption about data distribution, and they are usually very fast to compute. In image classification, the decision trees are mostly reliable and easy to interpret, as their structure consists of a tree with leaves which represent class labels, and branches that use logical conjunction to produce a value based on an 'if-then' rule. These values produce a set of rules that can be used to interpret the instances in a given class." This was proved viable, and reached up to an 85 percent accuracy rate upon using this method.

Another way to classify images is by using statistical algorithms, such as Naive Bayes or K-Means. According to [6], data scientists "applied a Naive Bayes algorithm in order to classify images based on the features from the edge histogram. A good accuracy was obtained by this application of image classification, and the approach is considered suitable for the automatic classification" of images. Additionally, in studies done by R. Rollet, et al. [8] they "implemented an image classifcation algorithm based on the radial basis function (RBF) neural networks combined with the technique of K-means." Their study showed that a k-means based image classification could achieve 94 percent success and remarked "Encouraging results were obtained from the classifcation of optical images as well as SAR images. The main properties of this algorithm are its self-organization, its universality and the fact that a priori knowledge on the images is not required."

The final and most basic way that images may be classified is by pixel-by-pixel search. According to D. Lu, et al. [3], "Traditional per-pixel classifiers typically develop a signature by combining the spectra of all training-set pixels for a given feature. The resulting signature contains the contributions of all materials present in the training pixels," which allows classification by comparing to the signature. This method alone doesn't hold much value, as according to [3], "insufficient, non-representative, or multimode distributed training samples can further introduce uncertainty to the image classification procedure," but can be used effectively as a helper in other methods, such as decision trees.

4 Modern Uses of Image Classification

There are many modern problems that have used image classification to help solve. One example is the use of census data in urban image classification. According to Victor Mesev [7], "Urban areas are undoubtedly one of the most challenging surfaces for image classification. Yet the dynamic nature of settlements means that the classification of urban areas has perhaps the greatest potential among the widest audience. For example, practitioners of urban monitoring, management, planning, and land-use zoning activities all need detailed information on the morphology, and especially functional use, of urban land at frequent time intervals. These needs have in part been met by image classifications which have allowed the consistent interpretation of the physical structure of urban land cover." [7] claims this strategy was applied to four dense housing settlements in the UK, and "the results show high site-specific accuracy, and improvements in class area estimates."

Image classification has been used to evaluate housing in the United States, as well. In fact, according U.S. patent US8775219B2, an image classifer used to classify rooftop condition for insurance invented by Aaron Swanson and Mark Helmlinger is "used to classify the condition of vegetation and property hazards. Classifying rooftop type and condition using remote spectral imaging greatly improves quantitative accuracy of roof inspections and decreases the cost of inspection across large geographic domains" and "the cost of residential property inspections and re-inspections performed via manual visual inspection is reduced."

While there are many uses of image classification now, according to Jana Machajdik, et al. [5] Psychology and Art Theory may be able to work in image classification as a way to determine and label emotion from images. They claim "we exploit theoretical and empirical concepts from psychology and art theory to extract image features that are specific to the domain of artworks with emotional expression." Using factors from Psychology and Art-Theory (like color or the Rule of Thirds), they were able to have a neural network determine an emotion that is fitting to the picture. While this technology is rather new, it goes to show how powerful image classification can be and how it could affect the future of AI.

5 Approach and Technology

While there are many options to choose from, we have opted to go with a CNN, aided by data augmentation and normalization. According to [4], "Given that spatial information can improve the classification accuracy, CNN utilises rich spatial features at multiple scales to describe the spatial structure of the data for classifying each pixel in the image. The rich complementary spatial details obtained at different scales helps boost the performance of the proposed classification method," whereas "SVM models alone may fail to extract features with different scales and to tolerate the large-scale variance of image objects."

Additionally, we opted to not use search trees or statistical algorithms to classify images and instead used a CNN due to the available methods (data augmentation and convolution) that will improve training efficiency and speed [2].

Because of this information, we have decided to approach this image classification by using a CNN and using data augmentation techniques, such as rotation and elastic warping, randomly across the training set, as well as grey-scaling each image. This approach, used by Patrice Simard [9], has "achieved the highest performance known to date on the MNIST data set, using elastic distortion and convolutional neural networks," by improving the quality of the data set via data augmentation and convolution. By using a CNN and augmented image data, we should be able to classify the flowers native to Minnesota.

A lot of neural network software currently exists. We will use Keras and TensorFlow (2.0.0) for Python to help develop and train the neural network classifier. In addition, we will likely use a web scraper to scrape images off of the internet that we can use as data to train the neural network. One potential option is to use a cloud/database management system to maintain the model and images if we have the need to.

However, while Keras and TensorFlow provide a good implementation of machine learning code, we will have to make the neural network design (layers, parameters, etc) from scratch, which will require testing once the dataset and code have been formed. This will be the largest portion of this project, as there are many different activation functions, optimizers and designs that can work and will need to be tested. According to [1], "DL-ReLU models are still comparable to, if not better than, the conventional Softmax-based DL models." Because of this, we have decided to use ReLU and Softmax for the basis of our network. We have decided to add two addition alterations as well: ReLU for the Convolutional layers and Sigmoid or tanh functions.

Some other additional software that we may need is data processing software. Python has an extensive library for this. We have decided to use Pandas, NumPy, SciPy and matplotlib in addition to TF and Keras to help form our data in a way that can be processed by our neural network.

To run our experiment, we will first gather image data on different plants then configure and design a CNN that is capable of learning and identifying different species of plants.

Afterwards, we will augment the data and save it by randomly assigning each image in the training set a different image augmentation in addition to adding a grey-scale filter to each image to keep the size of the CNN relatively small. The neural network will then be tested over the testing set.

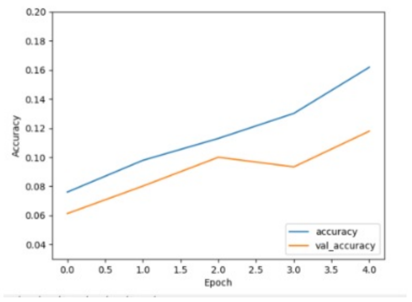
6 Experiment Design

In order to classify the images using a CNN, we must first determine the architecture and hyper-parameters that will give us the best results. We decided that we are going to train using 2 convolutional layer and 3 dense layers, one of which is the output layer. In addition we had to determine the size of each convolutional layer and what activation functions to use on each layer. To do this, we tested different configurations of the neural network over different numbers of epochs and compared the accuracy from each.

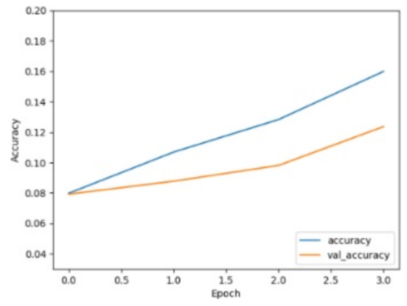
The first variable is convolutional layer size. We have decided on two options that we will test: (1) A 32-node convolutional layer, a 2D Max Pooling Layer, and a 64-node convolutional layer (2) A 32-node convolutional layer, a 2D Max Pooling layer, and a 128-node convolutional layer.

The second variable is the activation function. We have decided on three different options for the activation function: (1) Rectified Linear Unit (ReLU) (2) ReLU + Sigmoid for non-convolutional layers (3) ReLU + tanh for non-convolutional layers. We decided to use ReLU for all options because of its success seen in image classification [1]. In addition to this, we are using Softmax as the activation function for the output layer in all cases. Softmax here is the best because it assigns probabilities that the image is in a specific class, allowing us to classify images into the most likely category.

The notation ”32/64” ReLU would denote a A 32-node convolutional layer, a 2D Max Pooling Layer, and a 64-node convolutional layer and a ReLU activation function.



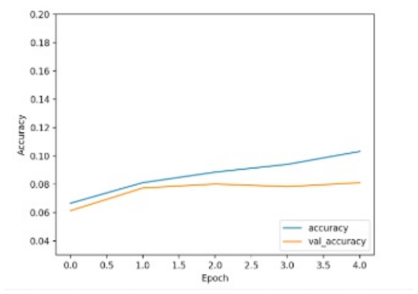
(a) 32/128 convolutional architecture over 5 epochs



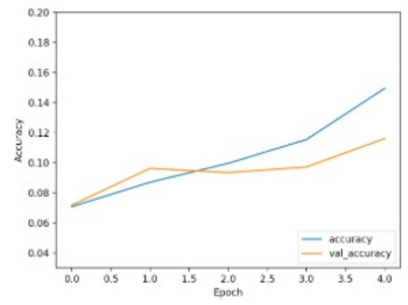
(b) 32/64 convolutional architecture over 5 epochs

Figure 1: ReLU training accuracy plot

Figure 1 is ReLU training over the two desired architectures. Overall model (a) achieved a validation accuracy of .1179 and model (b) achieved a validation accuracy of .1236. Even

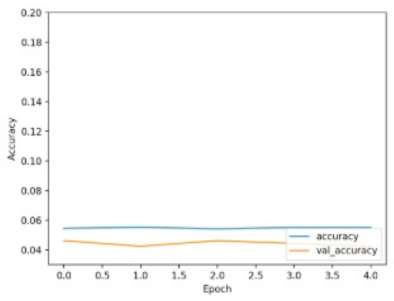


(a) 32/128 convolutional architecture over 5 epochs

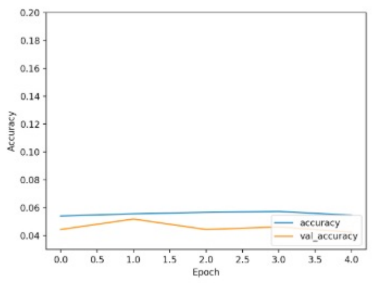


(b) 32/64 convolutional architecture over 5 epochs

Figure 2: Sigmoid/ReLU training accuracy plot



(a) 32/128 convolutional architecture over 5 epochs



(b) 32/64 convolutional architecture over 5 epochs

Figure 3: tanh/ReLU training accuracy plot

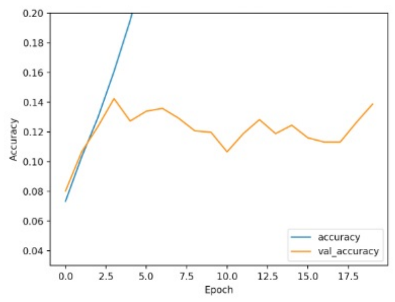
though they are similar, when using ReLU on the convolutional and dense layers, model (b) overall provides a more stable training process and the margin between training and validation accuracy was smaller on average in model (b).

Figure 2 is Sigmoid/ReLU training over the two architectures. Model (a) overall had little overfitting, but accuracy begins to flatten out near 3 epochs. Model (b) has relatively little overfitting as well, but learning continues at higher epoch levels. Once again because of this, model (b) performed better.

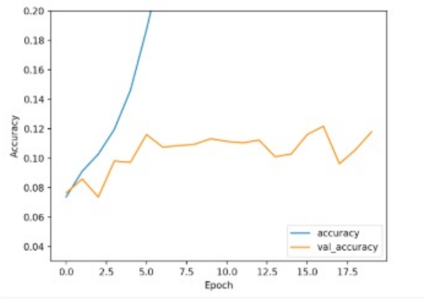
Figure 3 is tanh/ReLU training. Both tanh models proved to learn very little, sitting at about .05 accuracy, or just about random. Both models (a) and (b) showed little improvement in accuracy over small or large amounts of epochs and will not be considered further.

Based on these results, we decided to continue with the archictecture from model (b) - a 32/64 convolutional architecture. In order to choose between ReLU and ReLU + Sigmoid, we decided to run model (b) from both ReLU and ReLU + Sigmoid over 20 epochs and then compare accuracy to determine the best method of classifying the plants.

As seen in Figure 4, overfitting begins to become a serious problem for both models (a) and (b). In the ReLU-only model, we see severe overfitting, evident by the validation



(a) 32/64 ReLU convolutional architecture over 20 epochs

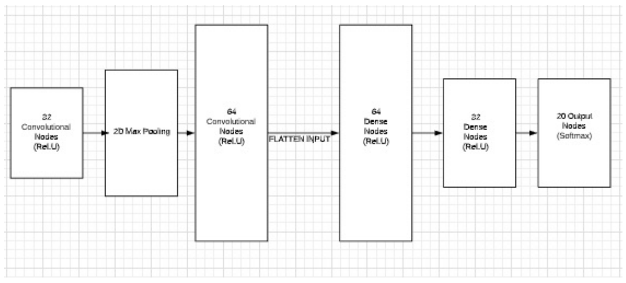


(b) 32/64 sigmoid convolutional architecture over 20 epochs

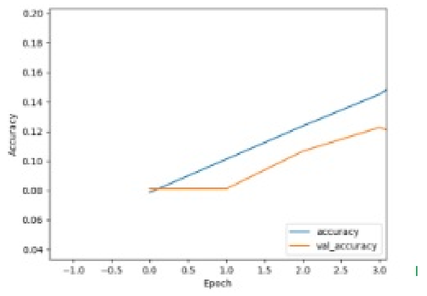
Figure 4: ReLU vs Sigmoid+ReLU

accuracy flattening and train accuracy still increasing, at about 4 epochs and had a peak accuracy of .1425. In model (b), we see overfitting from very early on at about 1.5 epochs and had a peak accuracy of .1217, but began overfitting at about .08. We have decided to use model (a) due to the fact that we are able to train more without overfitting, giving us better predictive capabilility. In addition, we will limit training to 4 epochs to prevent overtraining.

7 Analysis of Results


32 Convolutional Nodes (ReLU)
2D Max Pooling
64 Convolutional Nodes (ReLU)
FLATTEN INPUT
64 Dense Nodes (ReLU)
32 Dense Nodes (ReLU)
20 Output Nodes (Softmax)

(a) Final Neural Network Architecture



(b) Final ReLU 32/64 Model over 4 Epochs

Figure 5: Final Model and Results

The results of seen in Figure 5 above indicate that the neural network was able to pick up on patterns and identify Minnesota wildflowers at a rate about 2.5x (p=.1226) higher than random guessing (1 out of 20 wildflower classes). While this isn't nearly optimal, it still shows significant learning from the network that we created. Significance can be done by doing a t-test. The average random guess where p = .05 over 100 pictures would get on average 5 correct. With the model where p = .1226 it would be 12.26. If we assume the standard deviation for each is 15 percent of correct answers we can do the following t-test:

$$mean_1 = 5 \tag{1}$$

$$std_1 = .75 \tag{2}$$

$$mean_2 = 12.26 \tag{3}$$

$$std_2 = 1.839 \tag{4}$$

$$T = \frac{12.26 - 5}{\sqrt{\frac{1.839^2}{100} + \frac{.75^2}{100}}} = 36.55 \tag{5}$$

$$\alpha = .05 \tag{6}$$

$$p = 0.00001 < \alpha \tag{7}$$

Based on this, we can clearly see that the neural network vastly and significantly out performs random guessing, which is expected in this case. However, the results are far lower than what we had initially imagined when starting this project. Many advanced CNNs used for image classification can reach success rates of 95+ percent for easy classifcations and can still reach 85+ percent for more difficult problems.

Additionally, overfitting was constantly an issue for us. As seen in Figures 1-5, overfitting often began early and stuck around for the remainder of training. We attempted to remedy this by adding extra data points of augmented data, but overfitting still remained in early epochs. We chose ReLU 32/64 (Figure 4a) because overfitting occurred later on in the process than the rest, which gave the best accuracy results.

Another issue we faced that impacted our results was limitations in data. Our data pool overall was relatively small, at approximately 5000 images over 20 categories initially and nearly 18000 after augmentation. The lack of data in the training set could have potentially caused the overfitting mentioned previously. Additionally, a larger data pool would have given more image variation which would have allowed the neural network to do more meaningful learning over time.

These issues overall limited our models capability to accurately pick up on patterns. Overfitting causes the model to pick up on trends that exist in the training set, but may not exist outside of it. As a result, the model fails to accurately predict outside of the training set, which is very evident in Figure 4. The limitations in data also limited our model from being more accurate. Having more data gives variation to the data set and having such a limited quantity of pictures for each species (less than 1000) didn't give enough information for the network to accurately determine the flowers from each other.

Even though our model and data pool faced issues, overall the experiment proved to be a success, as the neural network was able to achieve better-than-random image classification through machine learning.

8 Conclusion

We attempted to classify Minnesota wildflowers using a convolutional neural network. We tested several architectures and activation functions, and we concluded that out of the tested sample a 32/64 ReLU CNN performed the best when it came to accuracy and avoiding overfitting. While the Sigmoid function performed reasonably well, it fell into overfitting multiple epochs before the ReLU model. The tanh model proved to do no learning on our dataset, as the validation accuracy never showed in significant increase over time.

In addition, by evaluating validation accuracy and train accuracy per epoch over 20 epochs, we were able to determine the ideal stopping point for trainging with a 32/64 ReLU network: 4 epochs. After this point, the model begins to overfit and validation accuracy stays the same as train accuracy increases.

When testing, the 32/64 ReLU CNN was able to achieve an accuracy of .1224, which is significantly higher than random guessing, which has an accuracy of .05. Even though the network did not reach optimal rates of successful classification, it still performed well enough to prove that the network was able to learn how to classify Minnesota wildflowers.

9 Future Work

In the future, we plan on getting this network to be working at a rate of at least .5 (to start off with). After the network becomes more successful, it could be possible to do a number of things with this network. For example, an application could be made that takes input from a user's camera on their phone, and then the picture is input into the network and information regarding the flower can be given out. This could possibly become like a scavenger hunt that gets people of all ages outside and walking around.

In order to get the network working better, we must address the data, preprocessing, architecture, augmentation and hyperparameters. Currently the data pool is relatively small, and we can work on getting more classes of wildflowers and total images to help improve the network. One constraint for using a neural network is the input size, and a big part of preprocessing is to make all the data the same size and shape. The way we accomplished this with different sized images was by squishing them into a square. This can make it difficult for the network because it changes the shape of the object in the photograph. To fix this in the future, another network could be used to intelligently crop each image to a square that includes the subject of each image. This could even be used for data augmentation; once an image is cropped, the remainder may be useful as another training example. Another part of preprocessing is the normalization of the data. This, too, could be achieved, intelligently, by another neural network to help distribute the values evenly and to help find import data points to add. The architecture was tested in this paper, but there are still many viable architectures that can be used in order to classify images, and further testing would certainly give us a stronger network. The data augmentation can be modified as well. Currently we only modify rotation, brightness, saturation and color. By adding other augmentations such as zoom, we can add to our dataset and help prevent overfitting and give us a more accurate model. The hyperparameters can also be tested. We tested only a small subset of possible activation functions and parameters and it is possible that there is one that we didn't test that would perform significantly better than what we did.

Another way to improve network performance is to have a program that builds the network and processes the data for you. This can come in many forms from building it from scratch to using Apple's recent Create ML application. Using an application like Create ML by Apple makes the process much more automatic and has the potential to create very good networks. However, It makes it hard to see what is going on under the hood and makes the network less customizable.

One additional piece of work we did for this project was allowing users to classify and image off of their computer using a command terminal. It takes in the picture, resizes it and inputs it into the model. The most likely plant classification is the prediction for that plant. So far I have only used it to classify pictures of my friends, but it can be used for more as well.

10 Contributions

Chanx411: Worked on the code for the neural network. Helped research methods and techniques for classifying images and helped develop and test the working model. Wrote majority of the introduction of the paper and helped write the experimental design, result analysis and conclusion.

Romxx014: Collected and labeled the data. Worked on the code. Researched methods for image classification. Did most of the testing for the experimental design section and helped write analysis and conclusions. Wrote section on intelligent solutions.

References

[1] F. M. Agarap. Deep learning using rectified linear units (relu). pages 1–7, 2017.

[2] T. M. K. Connor Shorten. A survey on image data augmentation for deep learning. pages 1–15, 2019.

[3] Q. W. D. Lu. A survey of image classification methods and techniques for improving classification performance. pages 1–20, 2005.

[4] H. H. et al. A comparison between support vector machine (svm) and convolutional neural network (cnn) models for hyperspectral image classification. pages 1–11, 2019.

[5] A. H. Jana Machajdik. Affective image classification using features inspired by psychology and art theory. pages 1–10, 2010.

[6] R. Jankovic. A survey on image data augmentation for deep learning. pages 1–9, 2007.

[7] V. Mesev. The use of census data in urban image classification. pages 1–8, 1998.

[8] W. L. S. W. . J.-M. B. R. Rollet, G. B. Benie. Image classification algorithm based on the rbinternational journal of remote sensing neural network and k-meansinternational journal of remote sensing. pages 1–7, 1998.

[9] P. Y. Simard, D. Steinkraus, J. C. Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, volume 3, 2003.

[10] A. A. D. C. D. S. Srdjan Sladojevic, Marko Arsenovic. Deep neural networks based recognition of plant diseases by leaf image classification. pages 1–20, 2016.