

University of Idaho

Digital Image Processing (CS 504-09)

Spring 2018 - Min Xian

Final Project Milestone 2

Adhar Partap Singh

[sing5830@vandals.uidaho.edu](mailto:sing5830@vandals.uidaho.edu)

Computer Science

# Task Description

This task is to develop machine learning models for classifying images of skin lesions. The HAM10000 dataset contains 10015 dermatoscopic images from various and unique groups. The images are Actinic keratoses and intraepithelial carcinoma / Bowen's disease, basal cell carcinoma, benign keratosis-like lesions (solar lentigines / seborrheic keratoses and lichen-planus like keratoses), dermatofibroma, melanoma, melanocytic nevi and vascular lesions (angiomas, angiokeratomas, pyogenic granulomas and hemorrhage). Our task is to design machine learning models and train them using the images and data to classify each lesion into one of these seven categories. The dataset can be found at Kaggle, Dataverse, or government science publication website (see references section).

# Resource Description

The resource we will use is Google Colab. Google Colab is a programming environment with many libraries installed for data, image processing, machine learning and other research. The most useful part is that the code is run in the cloud using powerful GPU and CPU which means we do not need a big computer to run complex models. We will be using Keras for building and training the models which uses TensorFlow for the low-level operations.

# Method Description

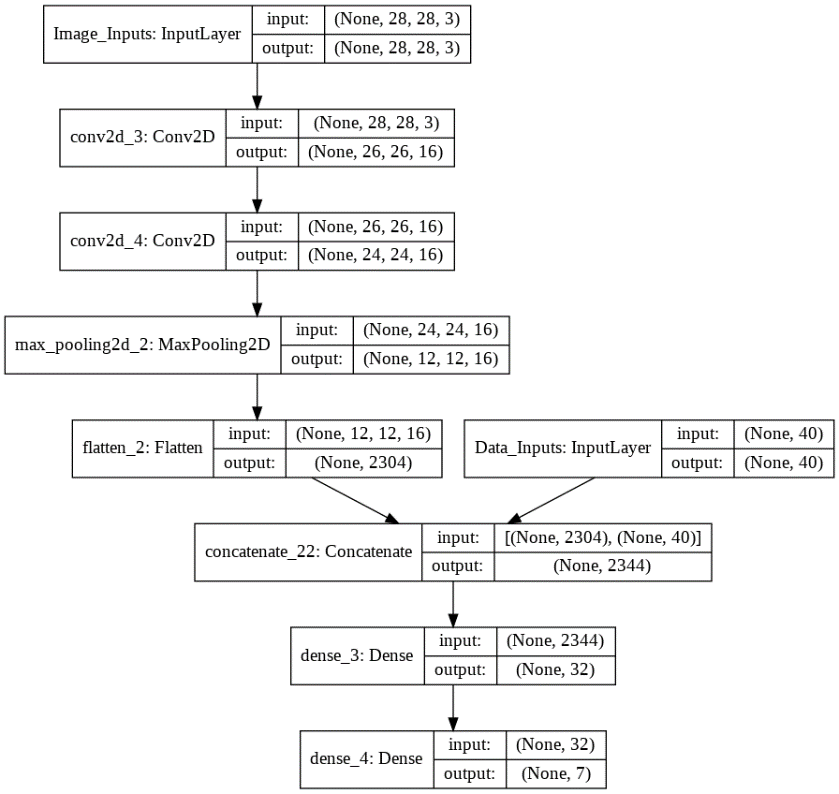
Because the dataset contains images and attributes, we will be using a combination of convolutional neural networks (CNN) and fully-connected deep learning neural networks. Our models will contain three main parts: the CNN for image processing, a neural network for data processing, and a neural network for joining the two into an output. This produces a Y-shaped model seen below. The python notebook running in Google Colab has seven main steps (see figure 1). First, the data is loaded from CSV into pandas DataFrames. Second, The data is pre-processed to convert attributes to categorical and to normalize all input between zero and one. Third, the 10015 entries are split into 70% for training, 20% for validation, and 10% for testing. Fourth, the models are defined using Keras calls. Fifth, the models are compiled using Keras calls. Sixth, the models are trained for 10 epochs on the training set. Seventh and finally, the models are evaluated on the testing set for accuracy and loss. During the training, accuracy and loss are kept in the history object allowing us to graph their improvement over time for our experimental results.

Figure 1: Program Flowchart

# Experiment Description

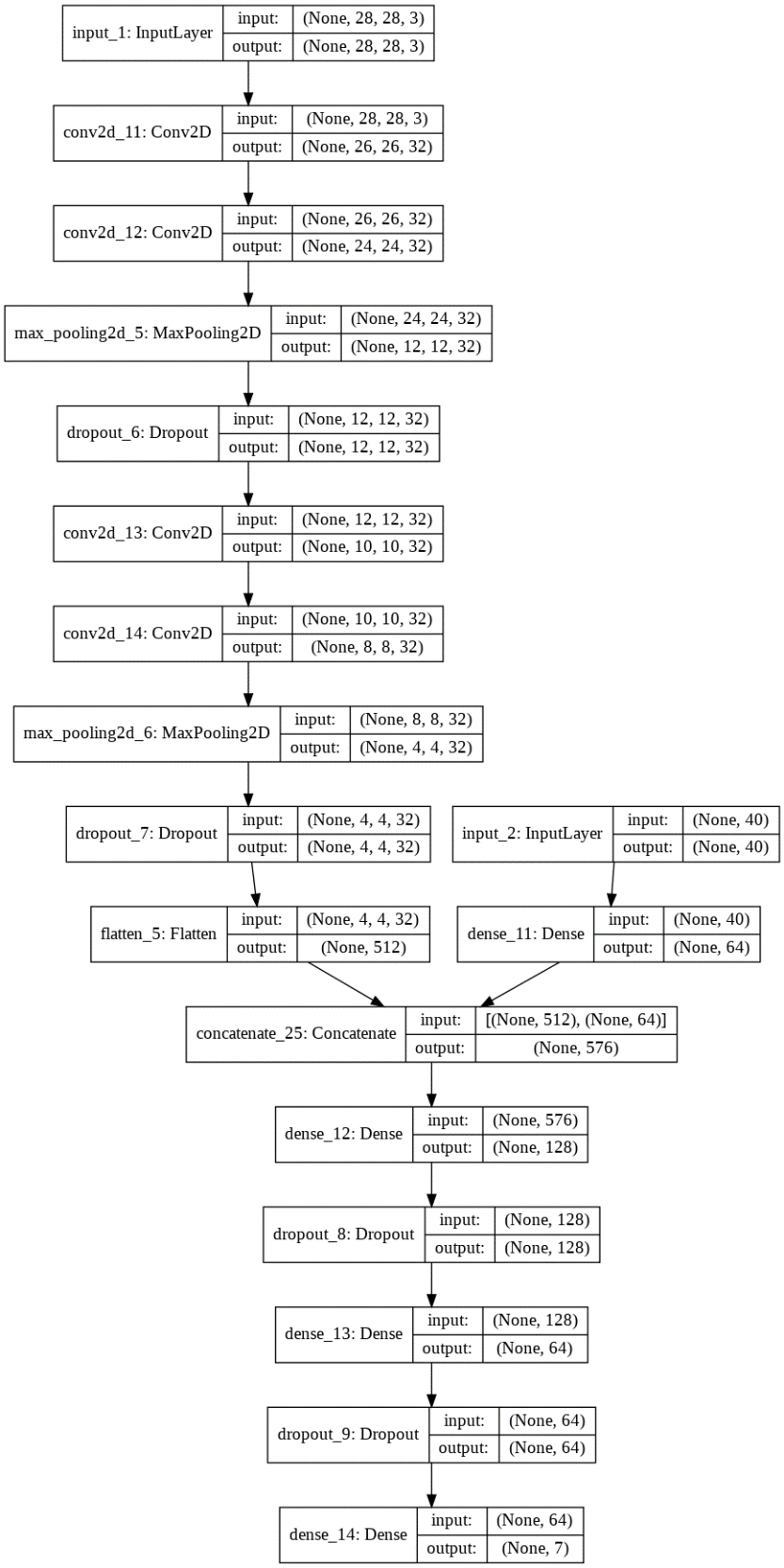
Six different models were used which tested two different variables: model complexity (low, medium and high) and whether or not we use dropout. The simplest model only uses two convolutional layers and two fully connected layers (see fig. 2). The most complex model uses four convolutional layers and four fully connected layers (see figure 3). Dropout is a regularization technique patented by Google that cuts communication between nodes randomly so they don’t learn to rely on any single input more than the rest. Dropout is known to help prevent overfitting. Visualizations of all six models are available in the jupyter notebook with the source code. This experiment was chosen when the prototypes were overfitting. They would get very high accuracy on the training set and very low accuracy on the testing set. The hypothesis was that dropout would help keep the accuracy on the testing set higher.

Figure 2: Simplest Model

# Experimental Results

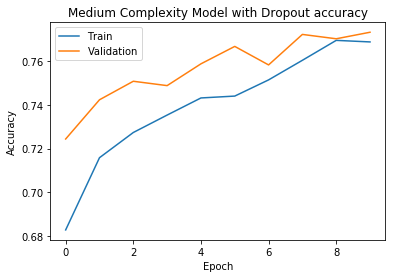
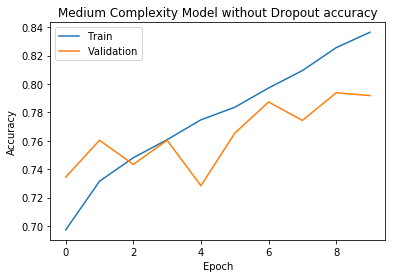
The models used have succeeded at classifying the dataset with around 70-80% accuracy. The programmers discussing the dataset on Kaggle have reported similar accuracies. The most interesting result was that the dropout increases accuracy on the testing and validation datasets (examples the model has not seen before) relative to the accuracy on the testing dataset. It also makes the testing accuracy lower. More complex models took more time to train but usually achieved higher accuracy on the training set, however, the complexity sometimes caused the models to overfit and do poorly on the validation and testing sets. Figures 4 and 5 show the accuracy on testing and training data at the end of training for each model. It is clear that even though dropout caused the training accuracy to be lower, it caused the testing accuracy to be higher. This is probably because the model is better at generalizing to new data. Let us now look at the accuracies during training. During training, the data is tested against the validation set, a set of data that has not been trained on. Figures 6 and 7 show the accuracy over time of the medium-complexity model with and without dropout. Here it is clearest what is happening. The dropout helps the validation accuracy remain high. Also, at the beginning the validation accuracy is higher than the training accuracy. This is because during training the dropout is enabled but during validation the dropout is disabled. Figure 8 shows all metrics and figures 9 shows confusion matrices for all models.

Figure 3: Most Complex Model

Figure 5

Figure 4

Figure 7

Figure 6

|  |  |  |  |
| --- | --- | --- | --- |
| **Low complexity with dropout** | Loss | Accuracy | Error Rate |
| Training set | 0.535 | 0.797 | 0.203 |
| Validation set | 0.62 | 0.763 | 0.237 |
| Testing set | 0.578 | 0.771 | 0.229 |
| **Low complexity without dropout** | Loss | Accuracy | Error Rate |
| Training set | 0.423 | 0.841 | 0.159 |
| Validation set | 0.603 | 0.777 | 0.223 |
| Testing set | 0.555 | 0.79 | 0.21 |
| **Medium complexity with dropout** | Loss | Accuracy | Error Rate |
| Training set | 0.506 | 0.803 | 0.197 |
| Validation set | 0.627 | 0.758 | 0.242 |
| Testing set | 0.581 | 0.775 | 0.225 |
| **Medium complexity without dropout** | Loss | Accuracy | Error Rate |
| Training set | 0.34 | 0.874 | 0.126 |
| Validation set | 0.606 | 0.765 | 0.235 |
| Testing set | 0.575 | 0.779 | 0.221 |
| **High complexity with dropout** | Loss | Accuracy | Error Rate |
| Training set | 0.511 | 0.797 | 0.203 |
| Validation set | 0.636 | 0.754 | 0.246 |
| Testing set | 0.577 | 0.77 | 0.23 |
| **High complexity without dropout** | Loss | Accuracy | Error Rate |
| Training set | 0.372 | 0.867 | 0.133 |
| Validation set | 0.617 | 0.757 | 0.243 |
| Testing set | 0.562 | 0.774 | 0.226 |

Figure 8

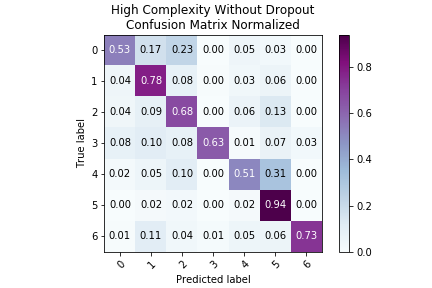
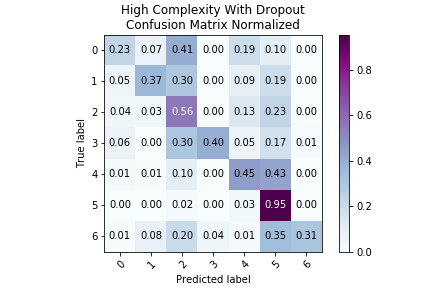
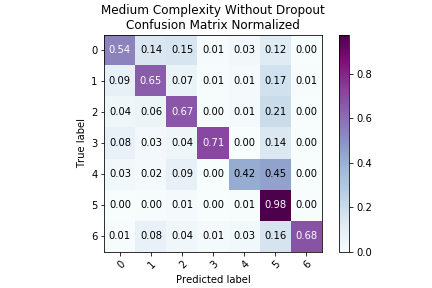
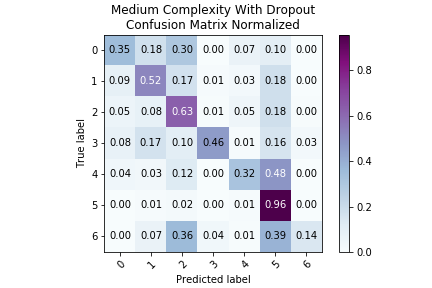
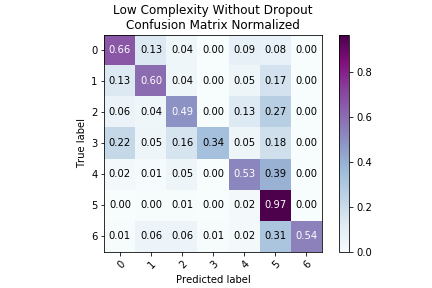
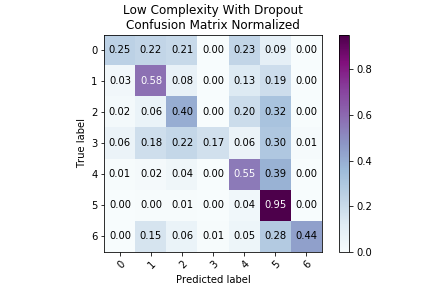


Figure 9

# Conclusions

Convolutional neural networks were a good choice for this problem because they achieved a good accuracy on the classification problem. The dropout experiment was a success because it showed clearly the difference in overfitting when using dropout. Complexity was not as conclusive because all networks behaved successfully, but medium complexity achieved the highest accuracy. Many image classification problems could be solved this way.

# References

Dataset at Kaggle:

<https://www.kaggle.com/kmader/skin-cancer-mnist-ham10000>

Dataset at Dataverse:

<https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T>

Dataset at the original publication website:

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6091241/>