Joseph Dobbelaar
Intro To AI Project 2
10/5/2023

Description of Experiments

Initial Model Configuration

The journey began with a baseline model provided in the template_ver2.py file. This model was relatively simple, consisting of a single Dense layer with just 4 units. The data augmentation techniques applied were minimal, with only a horizontal flip and a rotation of 0.3. The hyperparameters set for this initial model were:

- Batch Size: 64
- Image Size 30x30 pixels
- Epochs: 20
- Optimizer: SGD with a learning rate of 1e-2

Experimentation

The initial results prompted a series of experiments to improve the model's performance. The first significant change involved adjusting the Dense layer's units to 32, increasing the epochs to 50, and resizing the images to 50x50 pixels. This configuration yielded an accuracy of approximately 35%, albeit with a slightly longer training duration.

Seeking further improvements, the architecture was expanded to include two pooling layers and two convolutional layers. This change not only enhanced the model's accuracy to around 60% but also extended the training time. The image size was adjusted to 100x100 pixels, and the batch size was increased to 128. To optimize the learning process, the learning rate was fine-tuned to 5e-4, and the optimizer was switched to Adam.

However, this architecture led to some overfitting. To counteract this, additional data augmentation techniques were introduced:

- Random Zoom: 0.3
- Random Translation: 0.3
- Random Contrast: 0.3

Feature Scaling

Feature scaling was not applied during experimentation. The model relied on the inherent normalization provided by the data augmentation and the layers within the neural network.

Challenges and Observations

Throughout the experimentation process, certain challenges arose. Increasing the learning rate led to a plateau in accuracy at around 25%, likely because I got stuck in a local maximum. The switch to the Adam optimizer, combined with a reduced learning rate, proved beneficial. However, the introduction of additional data augmentation techniques initially resulted in reduced accuracy, necessitating fine-tuning of the augmentation parameters.

Additionally, in the pursuit of achieving higher accuracy, I initiated several training runs with a very high number of epochs. Two runs were set at 1024 epochs and one at 500 epochs. These runs were eventually canceled after about 200 epochs. The model wasn't showing any significant improvement, indicating that longer training wasn't necessarily beneficial and could potentially lead to overfitting.

Description of the Best Model / Training Procedure

Model Architecture

The best-performing model was designed with a combination of convolutional and dense layers. The architecture is as follows:

- Input Layer: The input layer accepts images of size 100x100.
- Convolutional Layers: Two convolutional layers were introduced.
- Pooling Layers: Two pooling layers were added after the convolutional layers.
- Dense Layer: A dense layer with 32 units.
- Output Layer: The final layer produces the classification results, assigning each input image to one of the flower categories.

Training Procedure

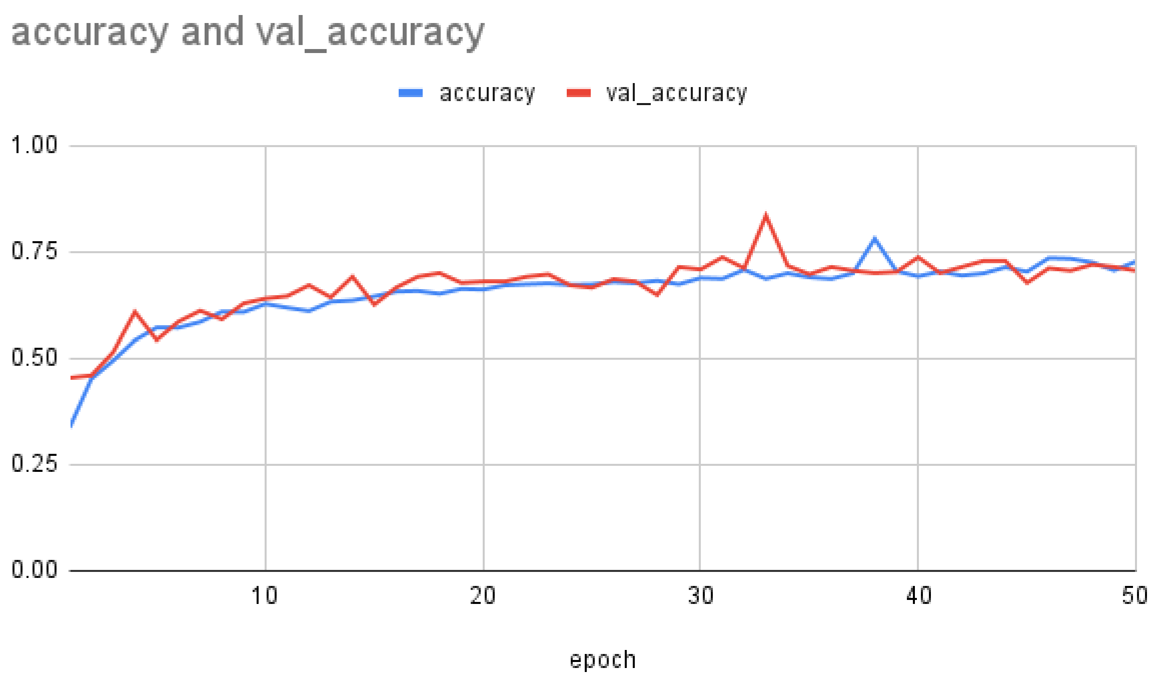
The model was trained using the following procedure:

- Data Augmentation: To increase the diversity of the training data and prevent overfitting, data augmentation techniques were applied. These included random flips, rotations, zooms, translations, and contrast adjustments. All of the augmentations were given a scale of 0.3.

- Optimizer: I used the Adam optimizer with a learning rate of 5e-4. I found that going any lower wasn't helpful, and higher than 1e-3 plateaued at 25% accuracy.
- Loss Function: The SparseCategoricalCrossentropy loss function was used, as per the template.
- Training Runs: The model was trained for 50 epochs. While some runs with a high number of epochs (1024 and 500) were attempted, they were halted around 200 epochs when it was observed that the model's performance plateaued. I wasn't too concerned about training for too many epochs, but I risked not turning this project in on time if I went any further than 50 during my last-minute experimentation.

Training Performance Plot

In the training performance plot, the model's accuracy and validation accuracy over the training epochs are visualized. This plot provides insights into how the model's performance evolved during training and can help identify potential issues such as overfitting or underfitting.



The accuracy and validation accuracy stayed close for the 50 training epochs, indicating that there was no over/underfitting. When I trained on 1024 epochs my accuracy was about 99% and by validation accuracy was closer to 60%. We plateaued at 75%, so I feel 50 epochs was sufficient.

Performance of the Best Model

Accuracy

The accuracy achieved was approximately 75%.

Precision

Here are the precision values from the confusion matrix:

- Daisy: 39/65 = 0.6
- Dandelion: 71/84 = 0.85
- Rose: 71/96 = 0.74
- Tulip: 63/93 = 0.68
- Sunflower: 41/46 = 0.89

We’re really good at dividing what’s a dandelion or sunflower, and generally poorer at daisies.

Recall

Here are the recall values from the confusion matrix:

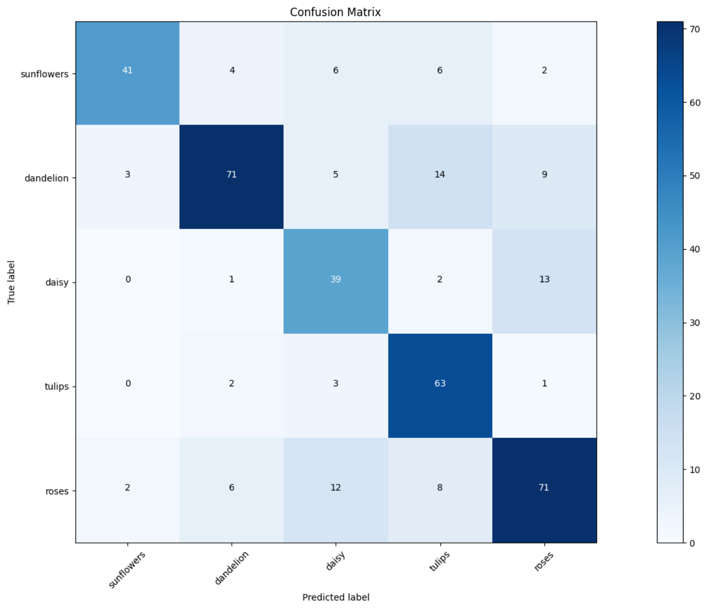
- Daisy: 39/55 = 0.71
- Dandelion: 71/102 = 0.67
- Rose: 71/99 = 0.71
- Tulip: 63/69 = 0.91
- Sunflower: 41/59 = 0.69

We often declare that something is a tulip! Otherwise, everything is pretty much 70% recall.

Observations

While the model demonstrates high accuracy overall, it has a strong ability to classify sunflowers and dandelions.It seems that, when unsure, it will be more likely to classify something as a tulip than any other flower.

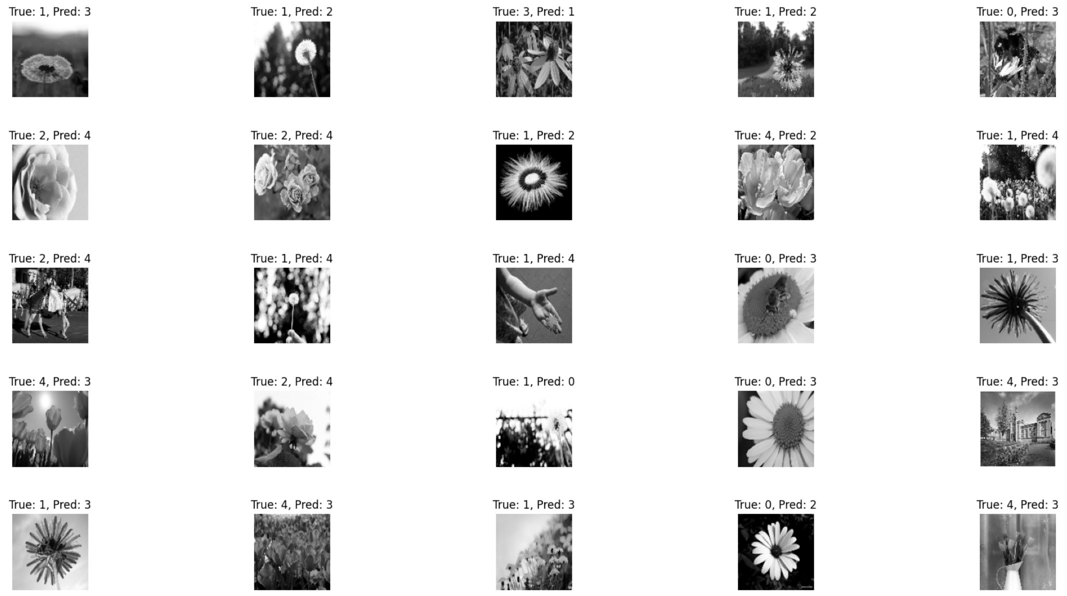
Confusion Matrix of the Best Model



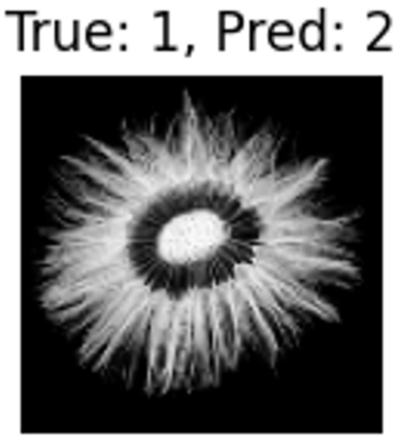
This confusion matrix shows that we're very unlikely to misclassify tulips, roses, and sunflowers, and dandelions, but daisies are harder to pick out.

When in doubt, we classify a flower as a tulip.

Visualization of Misclassified Images

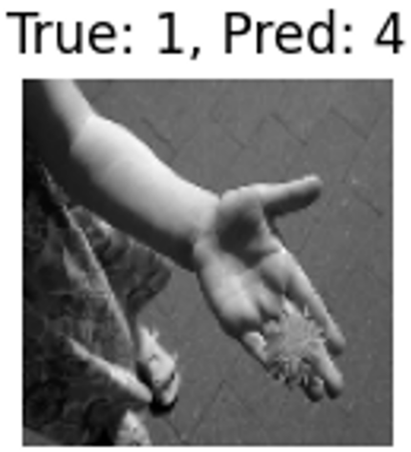


Firstly, here's 25 images that were misclassified. From this, I can tell that there are a lot of daisies. I think this is because daisies, in general, look like other flowers. There are three in particular that I'd like to look closer at:



This dandelion has had the top blown off, so it really looks like a daisy. I can understand why this would have been classified incorrectly.

This is just a picture of someone on a horse. I wouldn't be able to pick out the right flower either.



I think this one is hard to classify just because of how little real-estate the flower gets. In that way, this photo is mostly noise. Maybe it thought that the arm was a stem?

The Code

In my submission, there are two python files. joe.py is for training the model, and joe-analysis.py is for loading the model, creating the confusion matrix, and showing misclassified images.