Effects of Salting Techniques on Accuracy

Fariha Ahmed

Professor di Troia

CS 171 - Sec 01

6 December 2022

Table of Contents

Table of Contents 2

1. Abstract 3

2. Dataset & Methodology 3
    2.1. Dataset 3
    2.2. Methodology 3
        2.2.1. Models 3
        2.2.2. Salting Techniques 5

3. Results 6
    3.1. Experiment 1: Deep learning model, no salting techniques 6
    3.2. Experiment 2: CNN model, no salting techniques 6
    3.3. Experiment 3: CNN model, grayscale images 7
    3.4. Experiment 4: CNN model, rotated images 8
    3.5. Summary of Experiments 9

4. Conclusion & Future Works 10
    4.1. Conclusion 10
    4.2. Future Works 10

5. References 11

1. Abstract

In this project, I am researching the effect of different preprocessing salting techniques on a model that is trying to distinguish between a picture of a horse and a picture of a human. I will be using the Tensorflow 'horses_or_humans' dataset as my dataset.

I will conduct each experiment by first splitting the dataset into a training and test set. I will run a deep learning model and a CNN model on the dataset to see which model works best. I will use this run as a base case experiment. In the experiments afterwards, I will only use the model that returned the highest accuracy results. In two experiments, I will try two different salting techniques to manipulate or distort the images in some way as a form of data augmentation. The salting techniques I will use will be converting the images to grayscale and rotating the images by a random angle. After shuffling the preprocessed and salted images, I will then input the data into the model. The model will train with ten epochs each and generate an accuracy score. In the end, I will compare each experiment's results with each other and with the base case to see which salting preprocessing technique generated the highest accuracy score.

2. Dataset & Methodology

2.1 Dataset

The dataset I am using is the TensorFlow 'horses_or_humans' dataset. It is a large set of images that depict either a single horse or a single human. The subjects of each photo are depicted in various poses and lighting, so there is a good diversity of images to train with. There are 1027 samples in the training dataset and 256 samples in the test dataset.

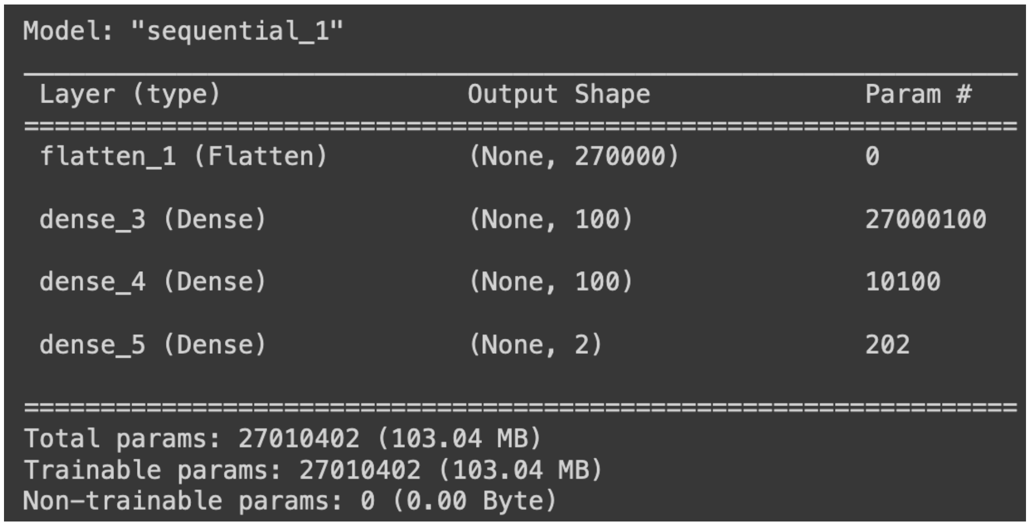
2.2 Methodology

To start off, I will load the dataset from TensorFlow and then split it into a training and test set. After, I will do basic preprocessing, which will consist of shuffling the data and batching it. I will also create some validation data to use when training the model.

2.2.1 Models

Then, I will try processing the dataset with two different models, a deep learning model and a CNN model. I will run both of the models on the dataset before any salting techniques are applied to the dataset. The purpose of trying two different models is to compare the two models and see which model returns better results. I will use the model with the highest accuracy score in my subsequent experiments. Additionally, this will serve as a base case since the models will be run on the dataset without any salting techniques applied. The results from this case will help compare the effectiveness of the different salting techniques by seeing if the subsequent experiments return a higher or lower accuracy score than this experiment.

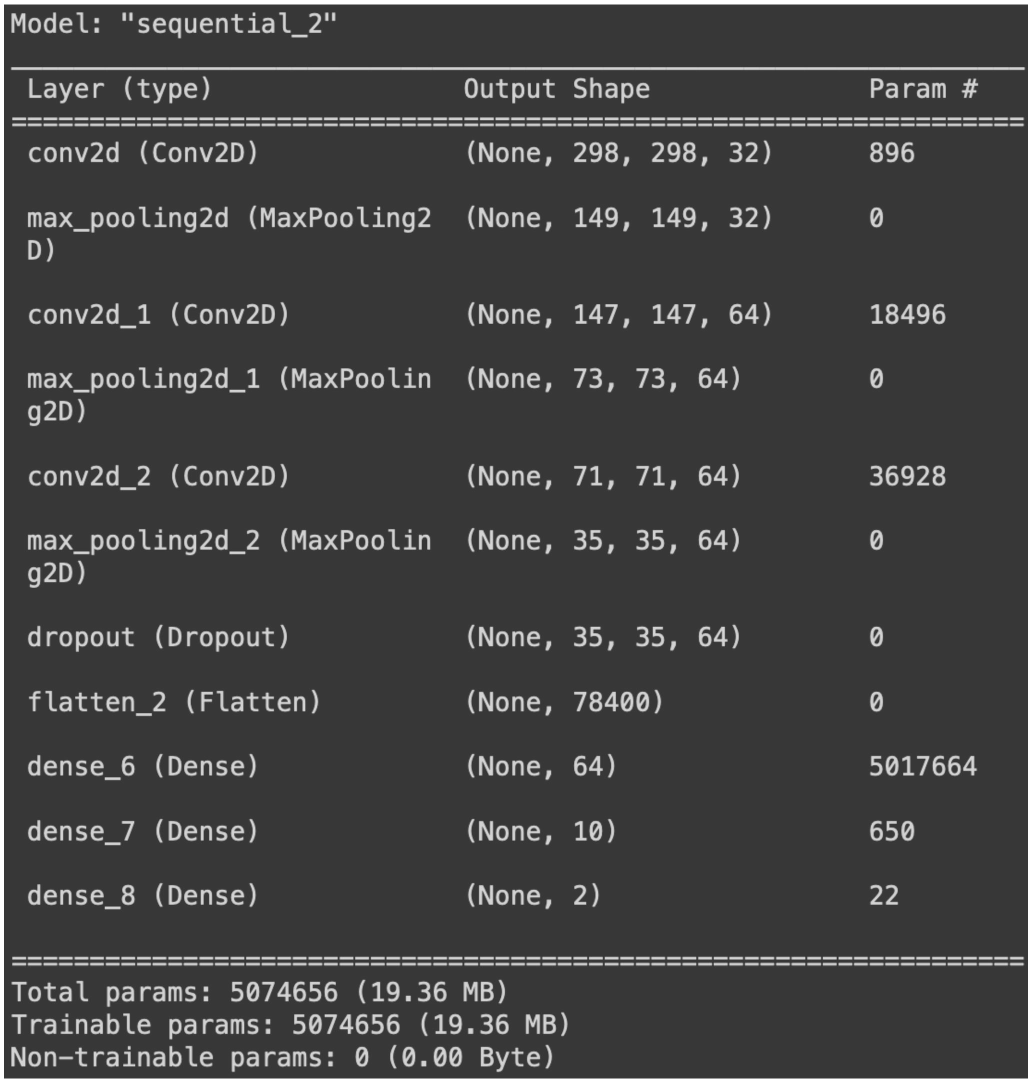
For the deep learning model, in addition to the preprocessing I have mentioned above, there will also be a scaling function to scale the images in the dataset. The model will have a flatten layer that will flatten the images, followed by three dense layers. Two dense layers will use relu as the activation function and have a hidden layer size of 100, and the last dense layer will use softmax as the activation function and have an output layer size of 2, since the dataset has two output classes.



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_1 (Flatten)         (None, 270000)            0

 dense_3 (Dense)             (None, 100)               27000100

 dense_4 (Dense)             (None, 100)               10100

 dense_5 (Dense)             (None, 2)                 202

=================================================================
Total params: 27010402 (103.04 MB)
Trainable params: 27010402 (103.04 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 1. *The architecture of the deep learning model*

For the CNN model, the dataset will not include scaling in preprocessing as the network is already preparing for the numbers to be flattened. The model will have three convolutional layers, all using relu as the activation layer, and three dense layers. The first dense layer will have a hidden layer size of 64, since it will take in the last output tensor from the convolutional base, which has a shape of (4, 4, 64). The second dense layer will have a hidden layer size of 10, and the last hidden layer size will have an output layer size of 2, since the dataset has 2 output classes. The CNN model will also include a dropout layer to help reduce overfitting.



```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 298, 298, 32)      896

 max_pooling2d (MaxPooling2  (None, 149, 149, 32)      0
 D)

 conv2d_1 (Conv2D)           (None, 147, 147, 64)      18496

 max_pooling2d_1 (MaxPoolin  (None, 73, 73, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 71, 71, 64)        36928

 max_pooling2d_2 (MaxPoolin  (None, 35, 35, 64)        0
 g2D)

 dropout (Dropout)           (None, 35, 35, 64)        0

 flatten_2 (Flatten)         (None, 78400)             0

 dense_6 (Dense)             (None, 64)                5017664

 dense_7 (Dense)             (None, 10)                650

 dense_8 (Dense)             (None, 2)                 22

=================================================================
Total params: 5074656 (19.36 MB)
Trainable params: 5074656 (19.36 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 2. *The architecture of the CNN model*

2.2.2 Salting Techniques

After I have chosen the model to continue my experiments with, I will conduct two experiments to test two different salting techniques. The first salting technique I will use is converting the image to grayscale since the images in the dataset are originally in full color. The second salting technique I will use is rotating the image by a random angle.

I will then input the dataset into the model and run the model. Afterwards, I will compare each experiment's results with each other and with the base case to determine which technique generated the best results. I will use accuracy evaluation metrics, so I will compare the model's results by their accuracy scores and losses. I will be using sparse categorical cross entropy as the loss function.

3. Results

3.1 Experiment 1: Deep learning model, no salting techniques

In this experiment, I ran a deep learning model with three dense layers on the unsalted dataset. I trained the model for 10 epochs. The validation loss did not increase at any point, so the model did not overfit. I got a test accuracy of 73.05% and a test loss of 5.65.

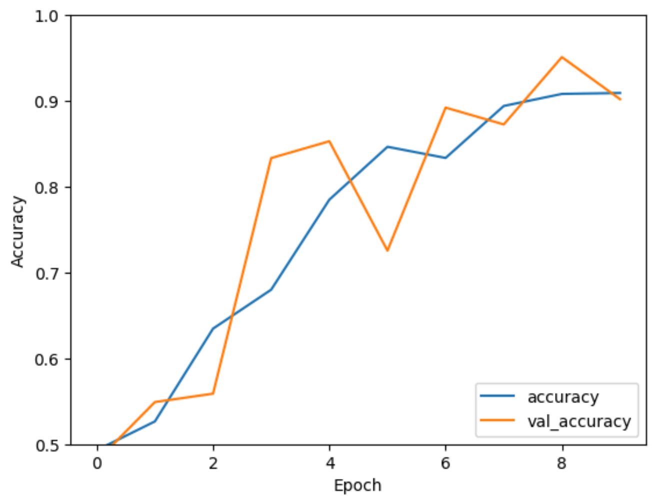


Figure 3. *The training and validation accuracies of the deep learning model over 10 epochs*

As seen in Figure 3, the training accuracy steadily increased and reached ~90% around the last few epochs, while the validation accuracy fluctuated a bit.

3.2 Experiment 2: CNN model, no salting techniques

In this experiment, I ran a CNN model with three convolutional layers and three dense layers on the unsalted dataset. I trained the model for 10 epochs. The validation loss did not increase at any point, so the model did not overfit. I got a test accuracy of 82.03% and a test loss of 3.35.

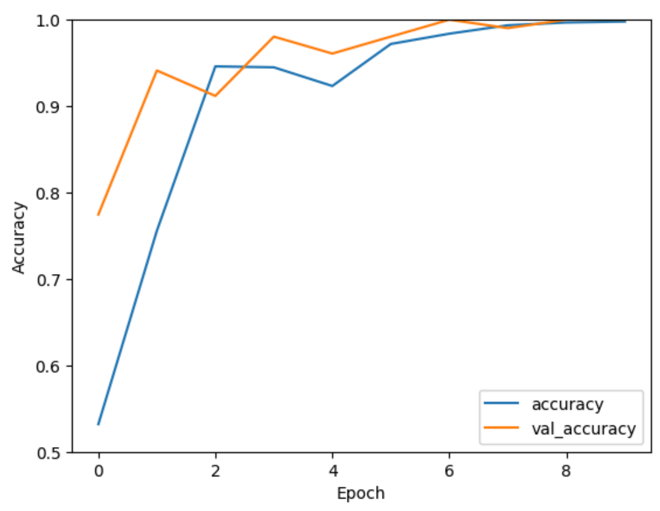


Figure 4. *The training and validation accuracies of the CNN model over 10 epochs*

As seen in Figure 4, the training accuracy and validation accuracy increased and reached 1.00% near the end.

Since the test accuracy for the CNN model was higher than the test accuracy for the deep learning model, I used the CNN model for my following experiments where the dataset was salted before the model processed it.

3.3 Experiment 3: CNN model, grayscale images

In this experiment, I ran the CNN model on the dataset after all its images had been converted to grayscale in preprocessing. I trained the model for 10 epochs. The validation loss did not increase at any point, so the model did not overfit. I got a test accuracy of 50.00% and a test loss of 0.69.

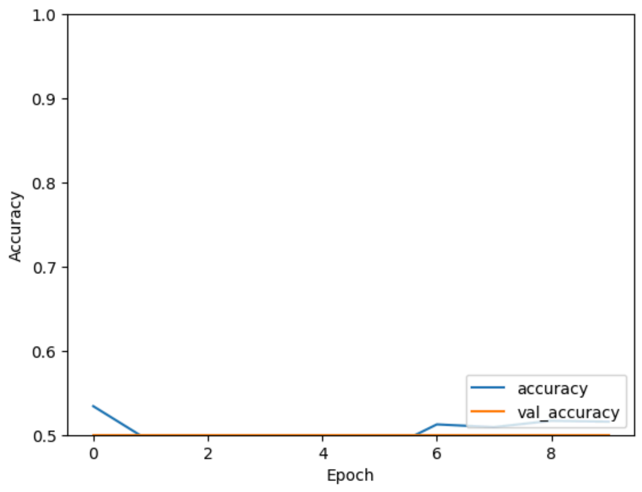


Figure 5*. The training and validation accuracies of the CNN model running on the grayscale images dataset over 10 epochs*

As seen in Figure 5, the training accuracy stayed around 50% for most of the epochs, and the validation accuracy did not increase at any point from 50%.

3.4 Experiment 4: CNN model, rotated images

In this experiment, I ran the CNN model on the dataset after all its images had been rotated at a random angle in preprocessing. I trained the model for 10 epochs. The validation loss did increase at one point, so the model did overfit a little bit. I got a test accuracy of 75.00% and a test loss of 1.85.

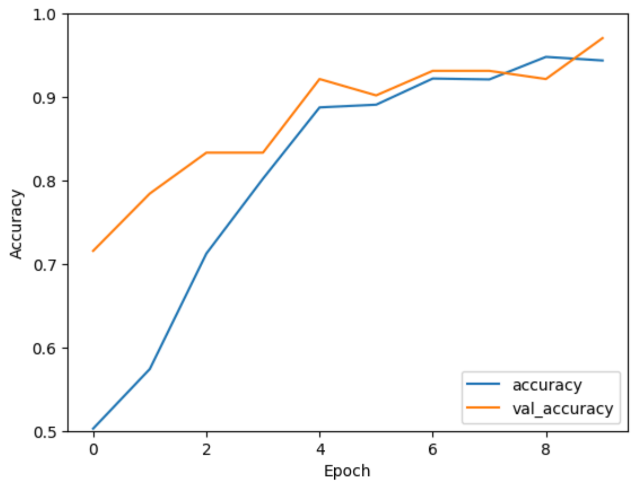


Figure 6*. The training and validation accuracies of the CNN model running on the rotated images dataset over 10 epochs*

As seen in Figure 6, the training accuracy increased during all 10 epochs and reached ~90% around the end, while the validation accuracy had a dip but still reached ~90% at the end.

3.5 Summary of Experiments

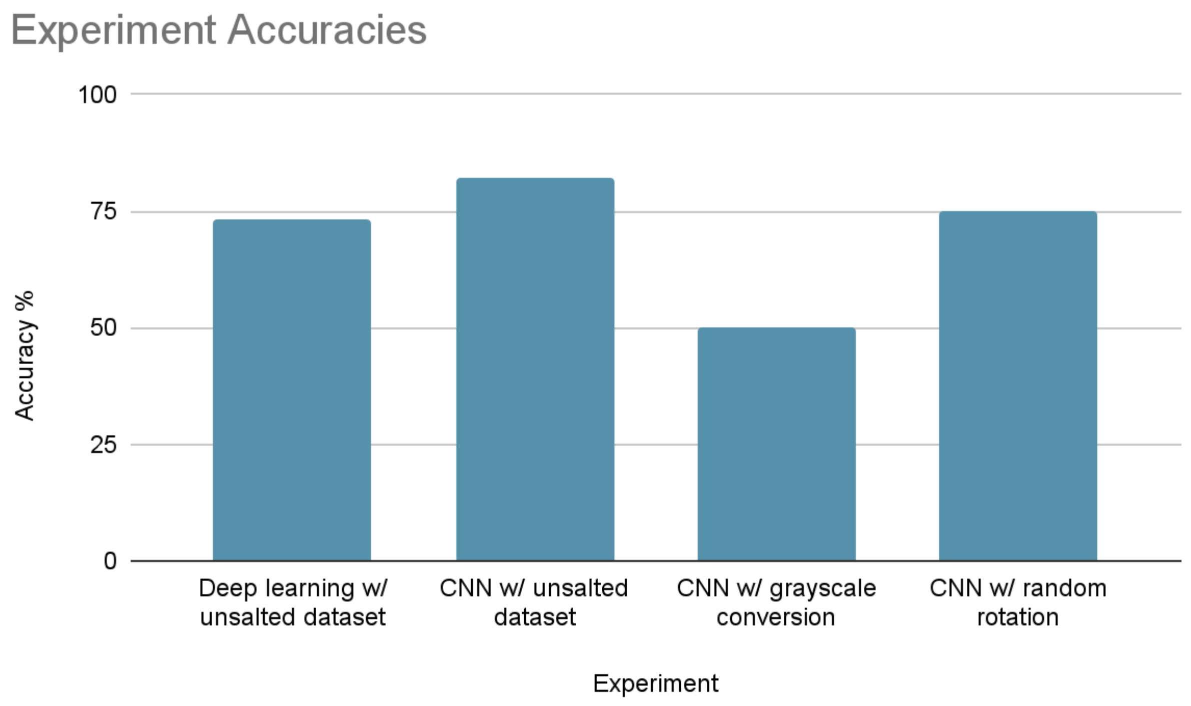


***Figure** 7. Column graph of the test accuracies from all the experiments*

***Table 1**. Summary of experiments and results*

#	Model Used	Salting Technique Used	Accuracy Score	Loss
1	Deep Learning	N/A	73.05%	5.65
2	CNN	N/A * base case	82.03%	3.35
3	CNN	Grayscale conversion	50%	0.69
4	CNN	Random rotation	75%	1.85

Out of all the experiments I ran, I found that using a CNN model on an unmodified dataset returned the highest accuracy score at 82.03%. The two salting techniques did not help the accuracy score become higher. In fact, when I salted the dataset by converting the images to grayscale, the accuracy score dropped to an alarmingly low score of 50%. When I salted the dataset by rotating the images at a random angle, the accuracy score was 75%, which was better than the grayscale conversion experiment and closer to the base case accuracy of 82.03%.

4. Conclusion & Future Works

4.1 Conclusion

Overall, the highest accuracy score I managed to get was 82.03%. Among all the experiments I tried, running a CNN model on an unsalted dataset returned the best results.

I set out on this project to see if adding salting techniques to the preprocessing of the dataset would help the model learn better. I tried the salting techniques of grayscale conversion and random rotation and both resulted in the model outputting an accuracy score that was lower than the accuracy score for an unsalted dataset. This could be due to the fact that data augmentation introduces more variability to the training data, making the data less clean and relevant. Thus, the learning problem for the model becomes more complex, and if it is not well equipped to handle this change, then the accuracy score will decrease.

4.2 Future Works

In the future, I would have liked to run more experiments that I did not have time to do during this project. One aspect I would have liked to explore is whether salting techniques used in a dataset that was run through a deep learning model could have returned different results. I would have also liked to try using different salting techniques in combination on the same dataset and then running the model on that dataset.

Something else I have liked to do was run more epochs for my experiments. I was only able to run 10 epochs per experiment as the model took a long time to run. I think increasing the epochs could have helped the models to learn better since the model can go over the data more times. I am curious whether increasing the epochs in my experiment that used a salted dataset could have helped increase the accuracy score.

Another thing I could experiment with in the future is varying the hyperparameters in the models used. For this project, I generally kept the hyperparameters the same and only changed the salting techniques used. I am curious as to whether adding more layers or increasing a hidden layer size in combination with salting techniques could have affected the accuracy score in any way.

5. Appendix

1. Code: https://colab.research.google.com/drive/1ovrFWzpTUfohXCrU4SVpmZ6GHK3eQwPm?usp=sharing
2. References
    a. Tutorial on how to use CNN: https://www.tensorflow.org/tutorials/images/cnn