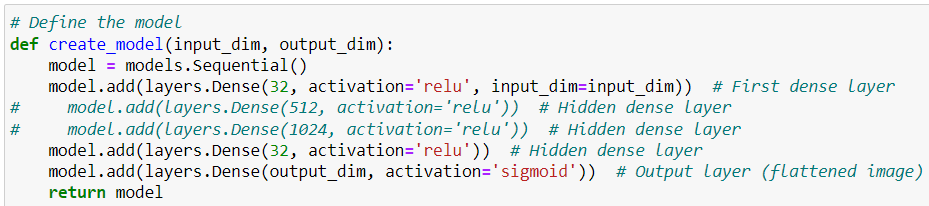
**Text to image compression**

**Data scenarios:**

10 images of random digits (28x28 pixels): 2.34 KB

10772 images of the same digit (28x28 pixels): 1.77 MB

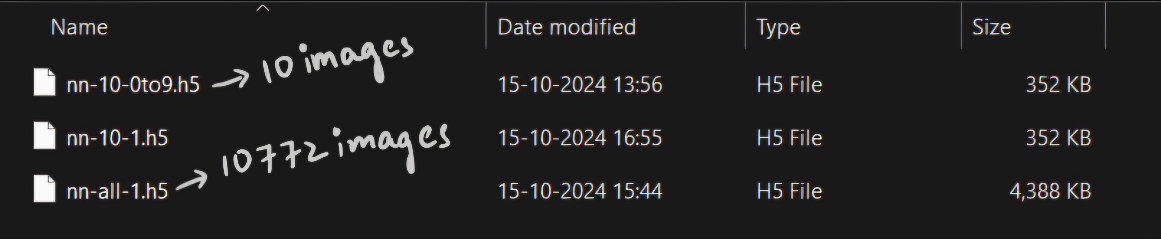
**Model:**



The model is a very basic Neural Network with the least amount of layers and nodes to decrease the size of the model as much as possible.

**Challenge 1:**

A typical model save file is generally big and increases in size as the training set(number of images) increases



This is because of the following reasons:

**1. Optimizer State:**

* Optimizers like Adam, RMSprop, etc., maintain internal states such as momentums, learning rates, and running averages, which are updated during training.
* These states grow in size as more updates happen, and the size of these states can depend on the number of epochs and the amount of training data.
* When you save the model using .h5 (or any other format that includes the optimizer's state), it saves these internal variables along with the model parameters.

**2. Training Metadata:**

* The .h5 file may also store metadata, such as loss, accuracy, learning rate, etc., during the training process.
* The amount of metadata might increase with more training steps (more data, more epochs), causing the file size to grow.

**3. Precision of Weights:**

If the model trains on more data, it might adjust its weights more finely. If you're saving weights with high precision (e.g., 32-bit floats), the saved file might grow slightly, especially if training is more extensive.

**4. Checkpoints/History:**

Some frameworks save additional training history, like the best weights at different points, depending on the settings you choose during training. This could also increase file size when saving a model trained on more data.

**5. Compression:**

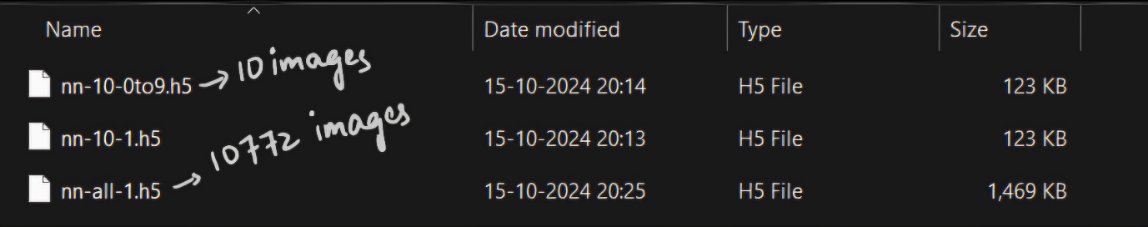
Depending on the dataset size and training process, saved weights might compress differently. This can vary between training runs, which could result in slightly different file sizes.

**Key Point:**

**The number of parameters (and thus the core weight size) remains constant**, but the additional information related to training (e.g., optimizer state) increases with the size of the dataset, causing the .h5 file size to grow.

**Solution:**

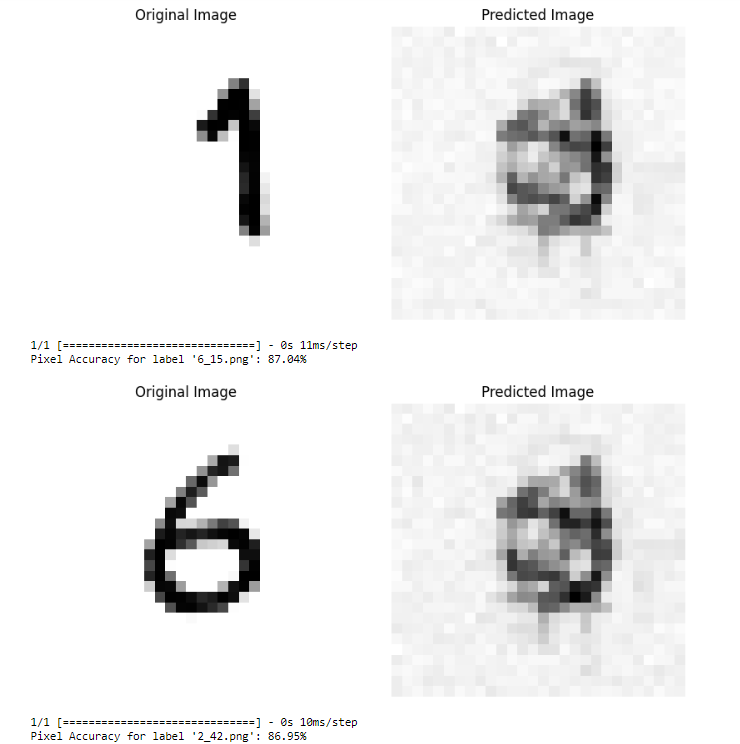
To overcome this, we can store the model weights and architecture only. This leaves us with almost a quarter of the typical model size, which reduces even further as the model gets bigger.



**Challenge 2:**

Running the model on small number of epochs gives inaccurate results.

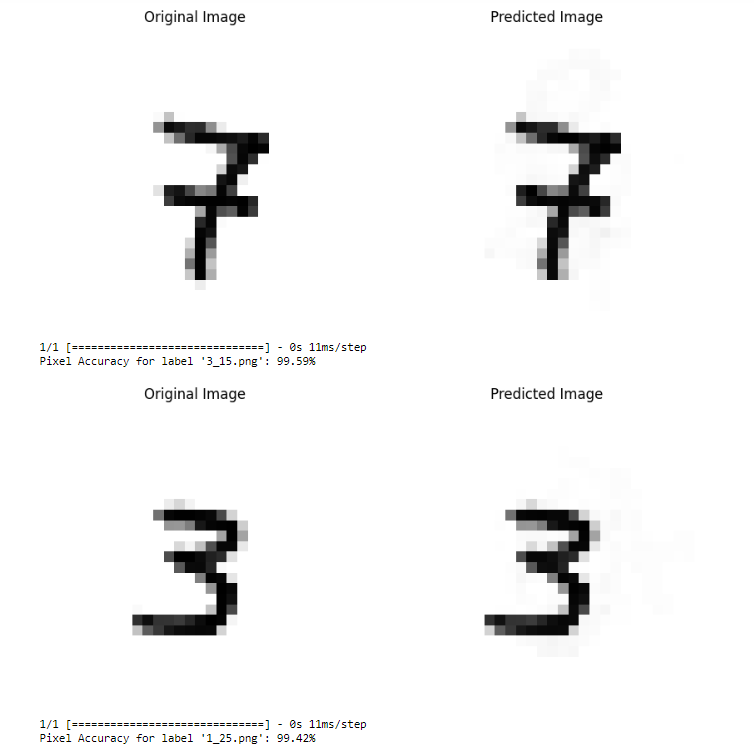
For example, in the below model ran with 100 epochs, it tries to create a mix of all images and predicts that weird mixed image for any given input



**Solution:**

To overcome this, we overfit the model, since overfitting the model gives exact output for the data it is trained on, which aligns with our goal. Below are predicts of same model with 1200 epochs.

As we run the model only once even if it takes a while to run it should be fine.

****

**Conclusion:**

Based on the above consideration, we achieved below results for 10772 grayscale images of 1

Original size: 18,60,720 bytes

Weights size: 15,03,456 bytes

The percentage decrease is approximately 19.20%

**Further Research and Experimentation:**

* We need to run more experiments on larger datasets, RGB images and other types of data.
* Experiment with model architecture and other ways to save model to try reduce the size even further.