**WIKI REPORT**

1. In this ICP I have learned how to perform Autoencoders on a given DataSet. And add two more layers to the encoder path and their corresponding two layers to the decoder path.
2. In this ICP the given task is to add two layers to the encoder path and their corresponding two layers to the decoder path.

In Encoder

* So, I have added LeakyReLU layers as it makes training faster.
* And second change i have added maxpooling2D as it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.
* And I have added 2 sets of conv2d and normalization with the higher filter values and removed the strides. Because the more layers are added the result is noise less.

In Decoder

* I have added the upsampling because in Encoder i used max pooling which downsamples the image that is why i used upsampling to upsamples the image
* I have used the LeakyRelu layer to make training faster.
* Here also I have added 2 more conv2d and batch normalization with an increased filter value because in the decoder it has to be exactly opposite of the encoder.

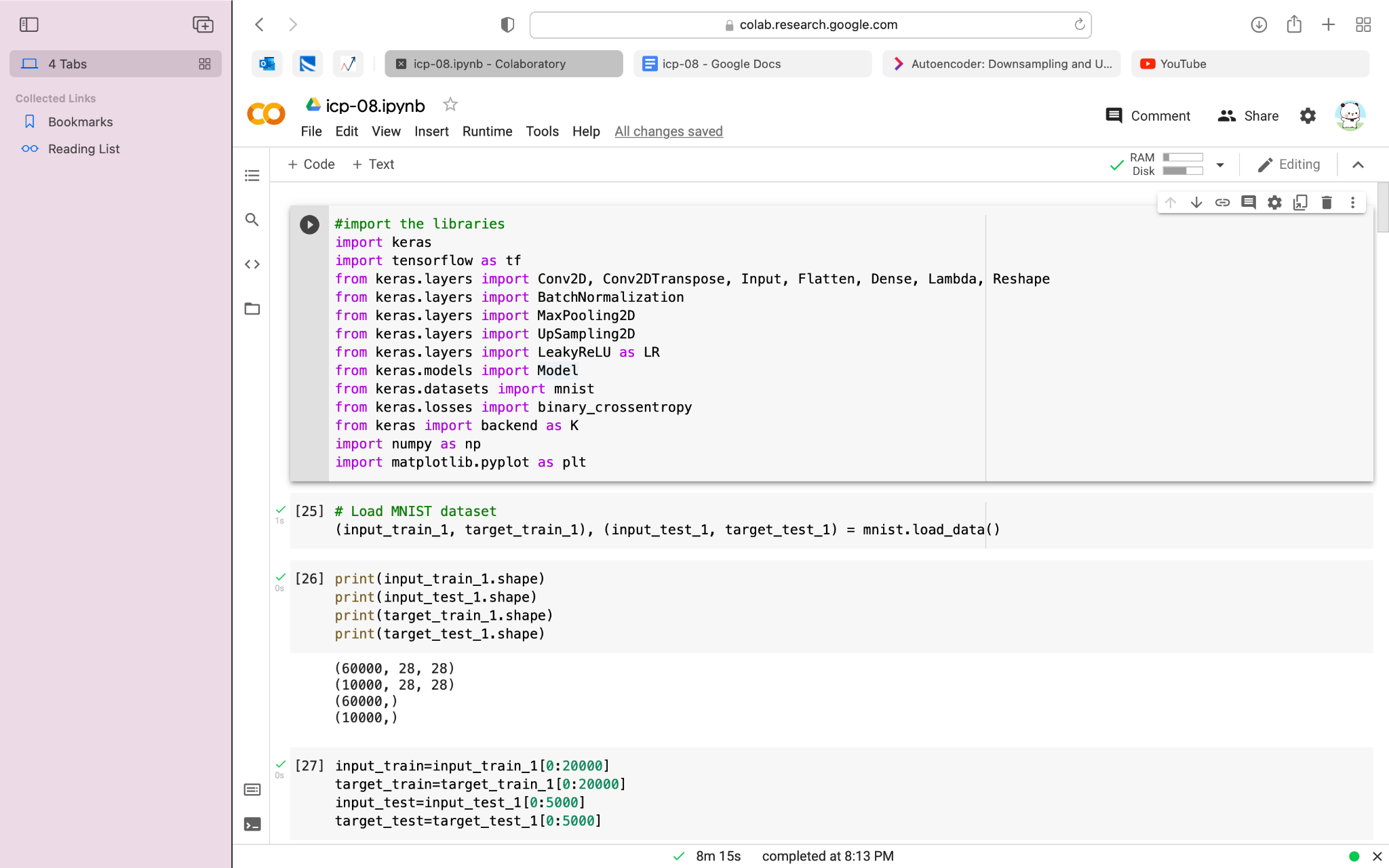
Explanation of ICP:

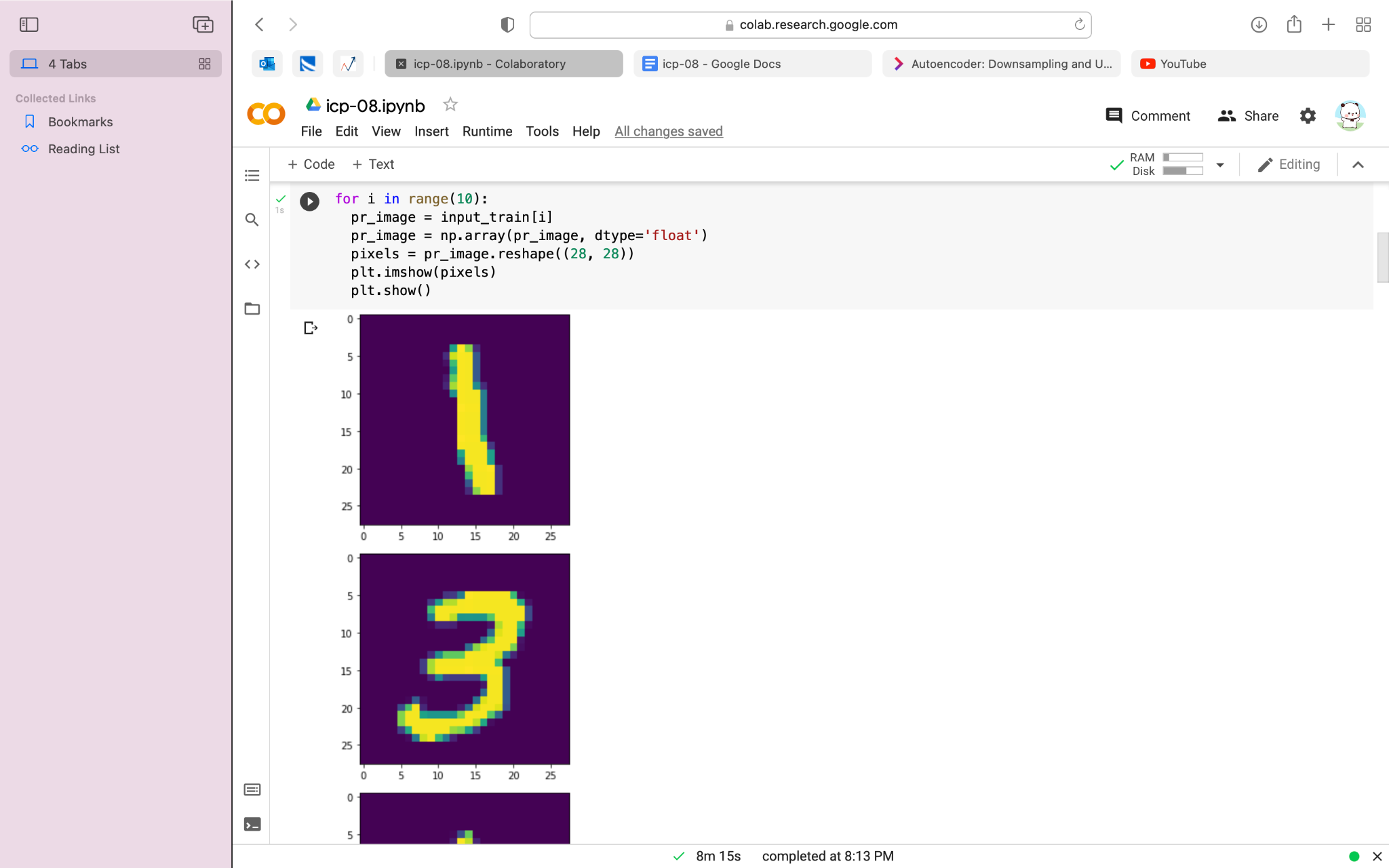
In this ICP i have done Autoencoder on MNIST dataset which contains different handwritten numbers. Autoencoder means it is a neural network that aims to copy their inputs to their outputs. They work by compressing the input into latent-space representation, and then reconstructing the output from this representation. Next I have reduced the size of the data as it's giving a memory error and requires more GPU ie.. I have reduced the training set from 60000 to 20000 and test set from 10000 to 5000. Then setting the configuration parameters for data and model. After that reshaped the data so that it takes the shape (X, 28, 28, 1), where X is the number of samples in either the training or testing dataset. We also set (28, 28, 1) as input\_shape. Next, I have parsed the numbers as floats, which presumably speeds up the training process, and normalizes it, which the neural network appreciates. Then Created the encoder. This is a three-step process: firstly, I have defined it. Secondly, I have performed something that is known as the reparameterization trick in order to allow us to link the encoder to the decoder later, to instantiate the VAE as a whole. But before that, I have instantiated the encoder first, as our third and final step. The first step in the three-step process is the definition of encoder. Following the connection process of the Keras Functional API, I have linked the layers together.

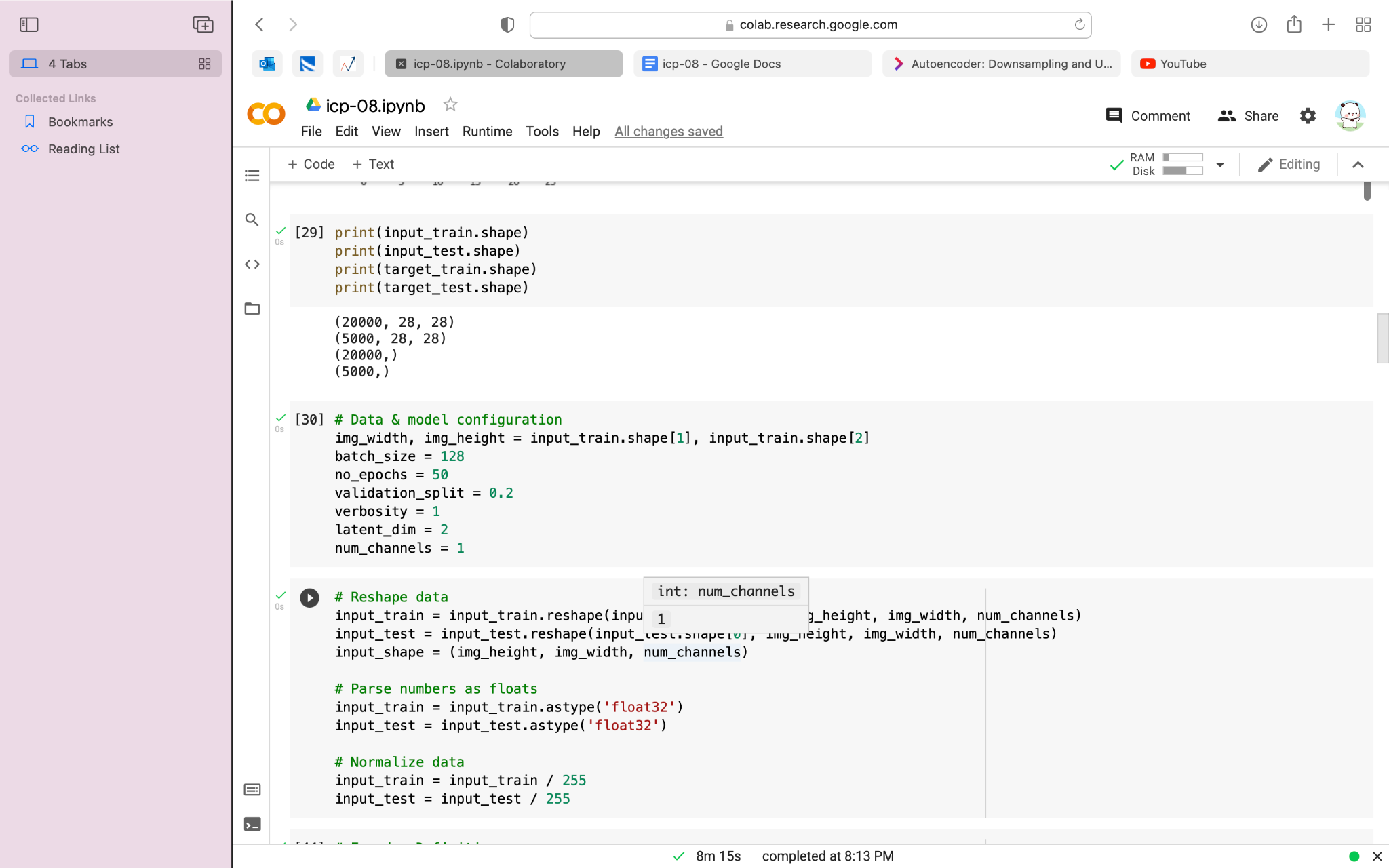
The first layer is the Input layer. It accepts data with input\_shape = (28, 28, 1) and is named encoder\_input. In the second layer I have added the LeakyReLU layer. As it speeds up the training.In the next layers i have used max pooling, conv2d and then added Batch normalization. Then defined sampling with a reparameterization trick. Used a reparameterization trick to ensure correct gradient. Instantiated encoder. Then decoded that code using conv2D, max pooling, upsampling and LeakyReLu. Instantiated Decoder. Created the whole VAE. Compiled and Trained. Then plotted the result.

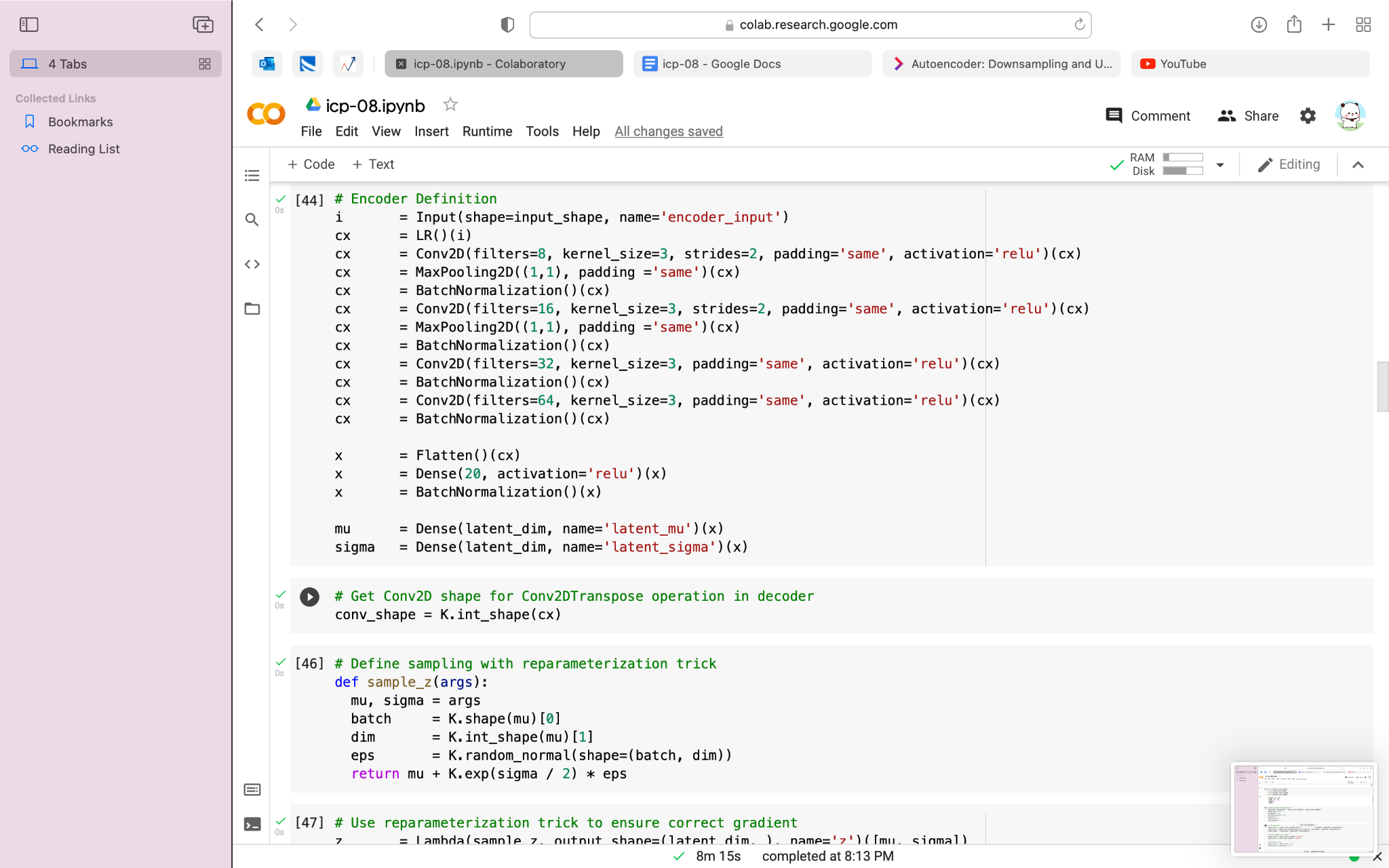
**C)**The challenges that I have faced in this icp is that the final output is noisy even though I have trained with a large amount of data. So, I added layers of conv2D and batch normalization with the higher filter values and the result was far better.

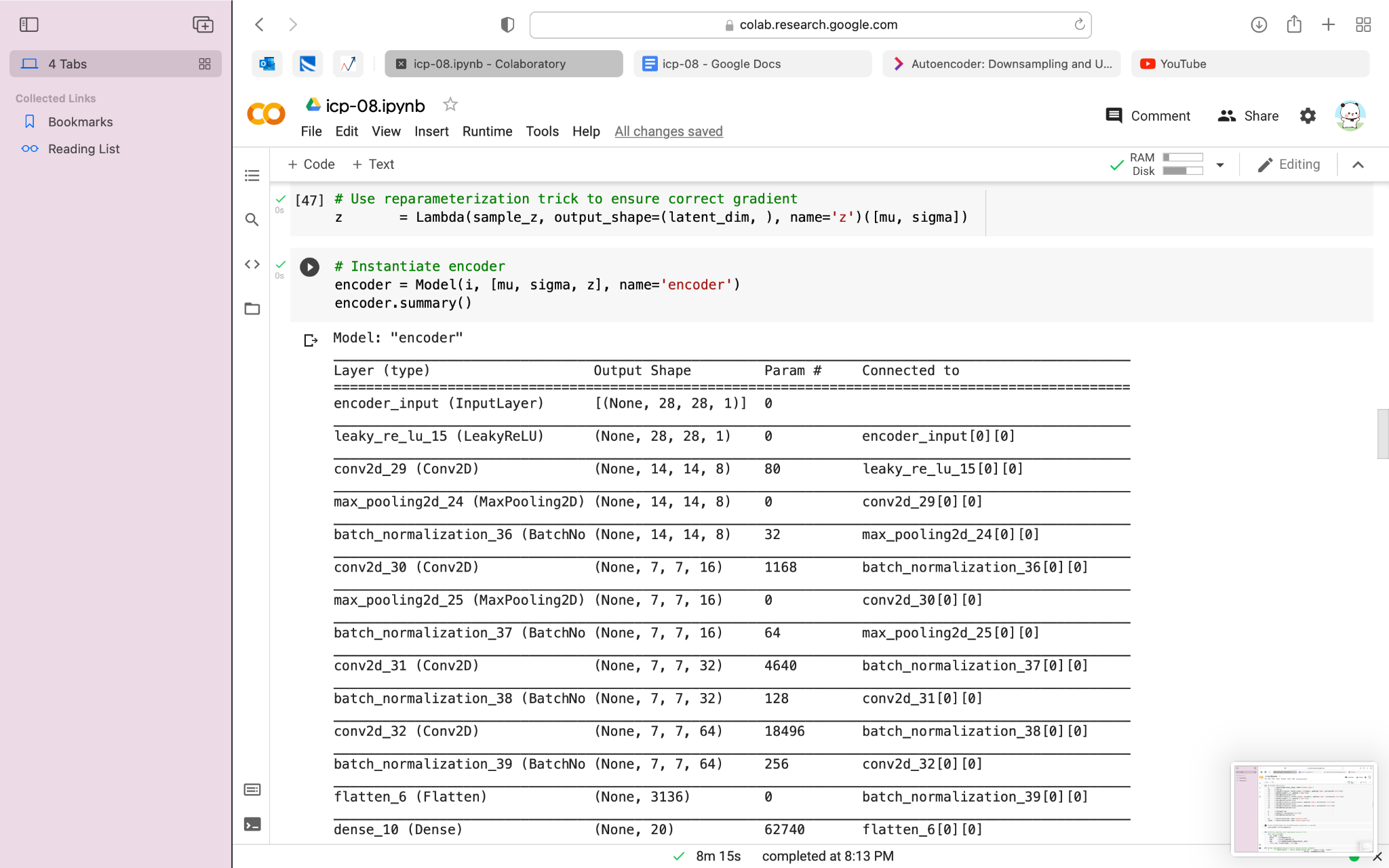
**D)Screen Shots:**

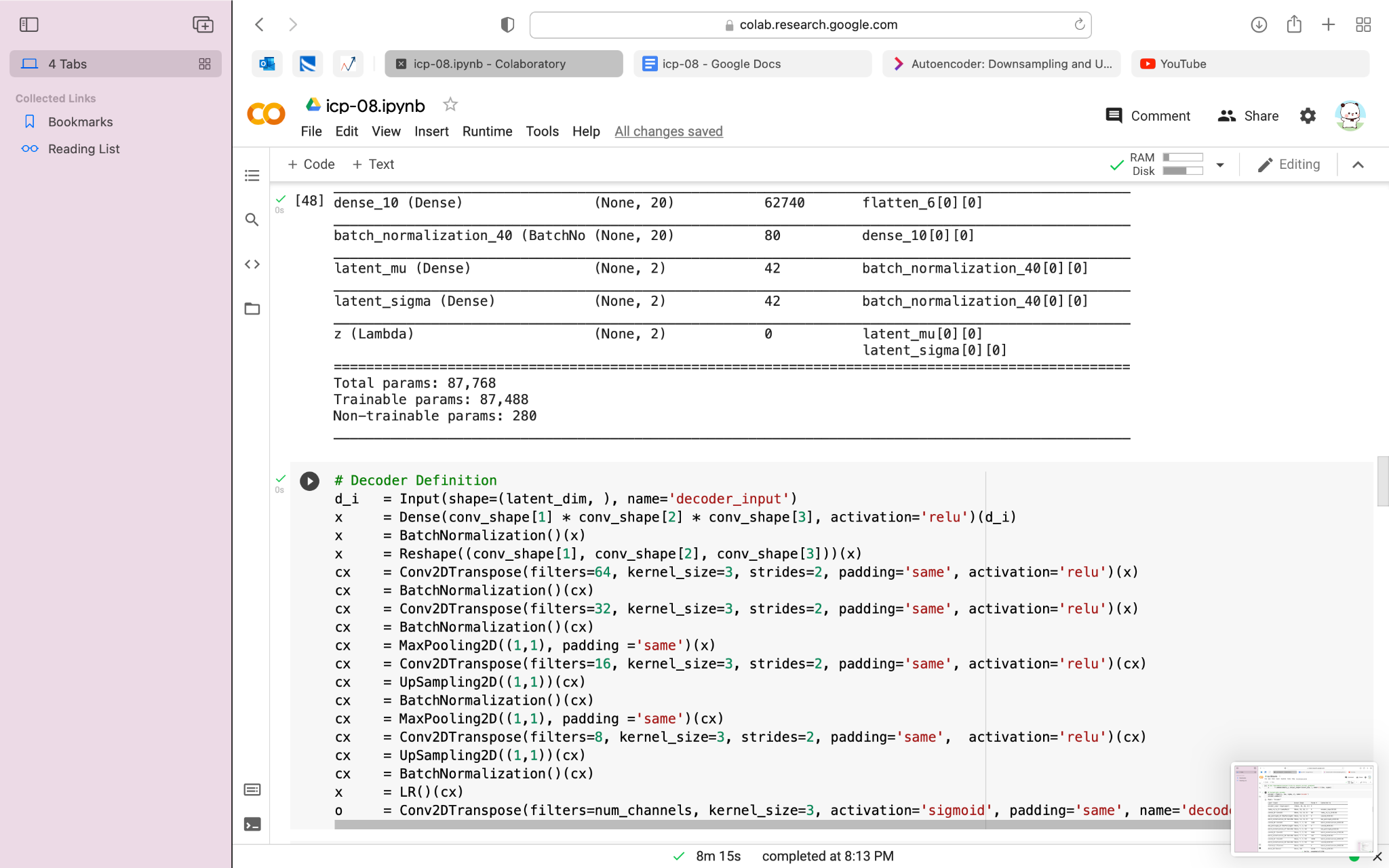
****

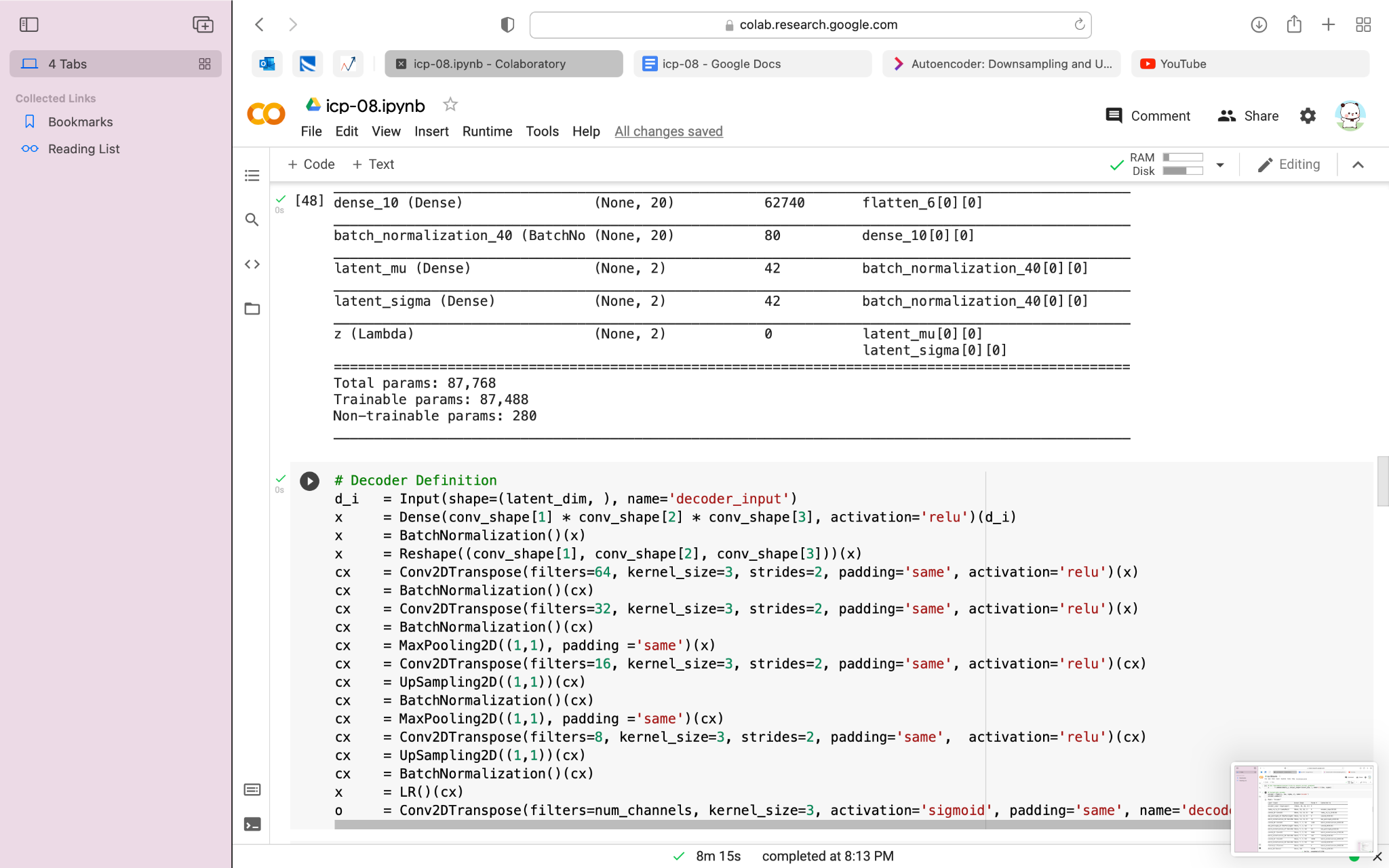
****

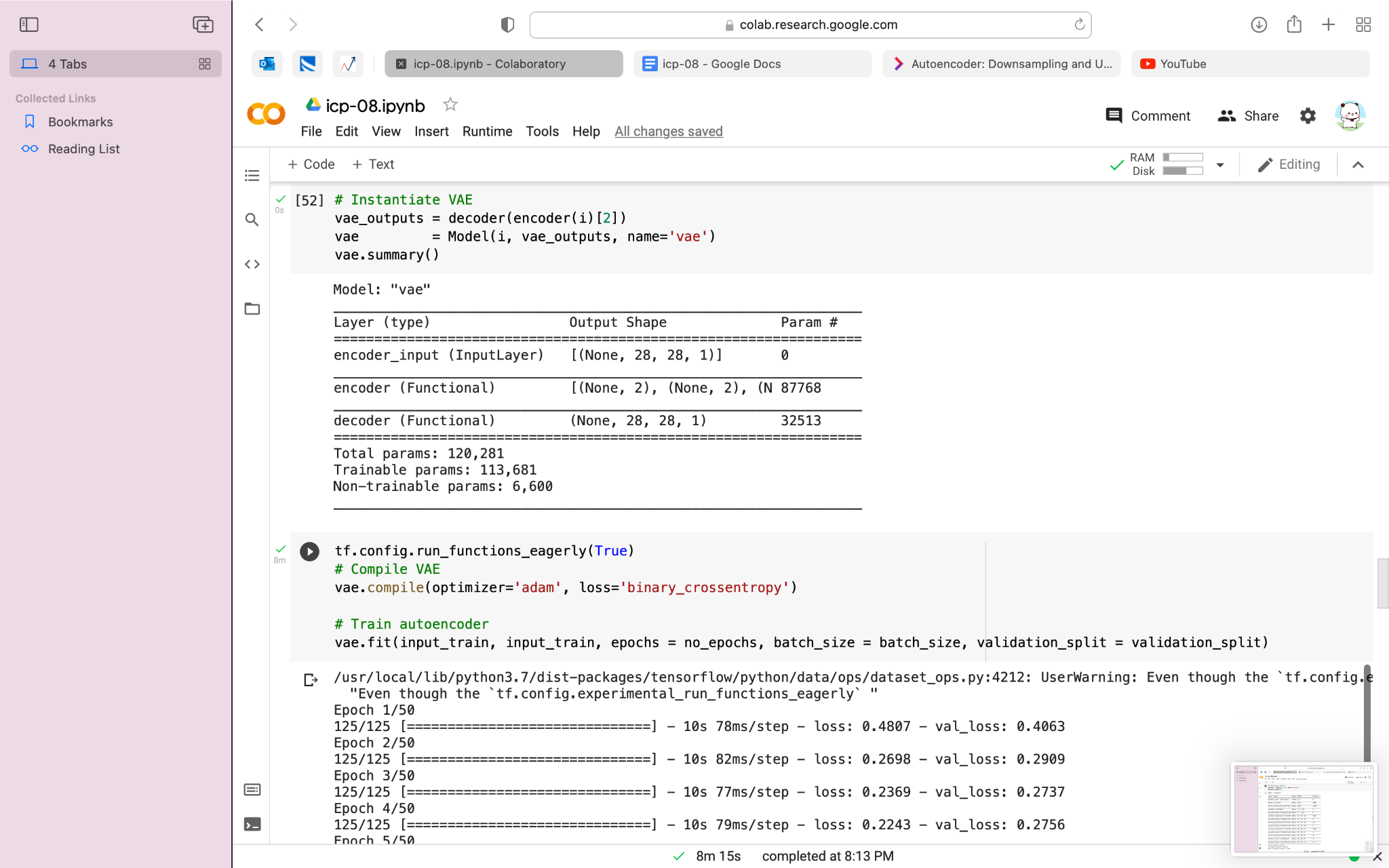
****

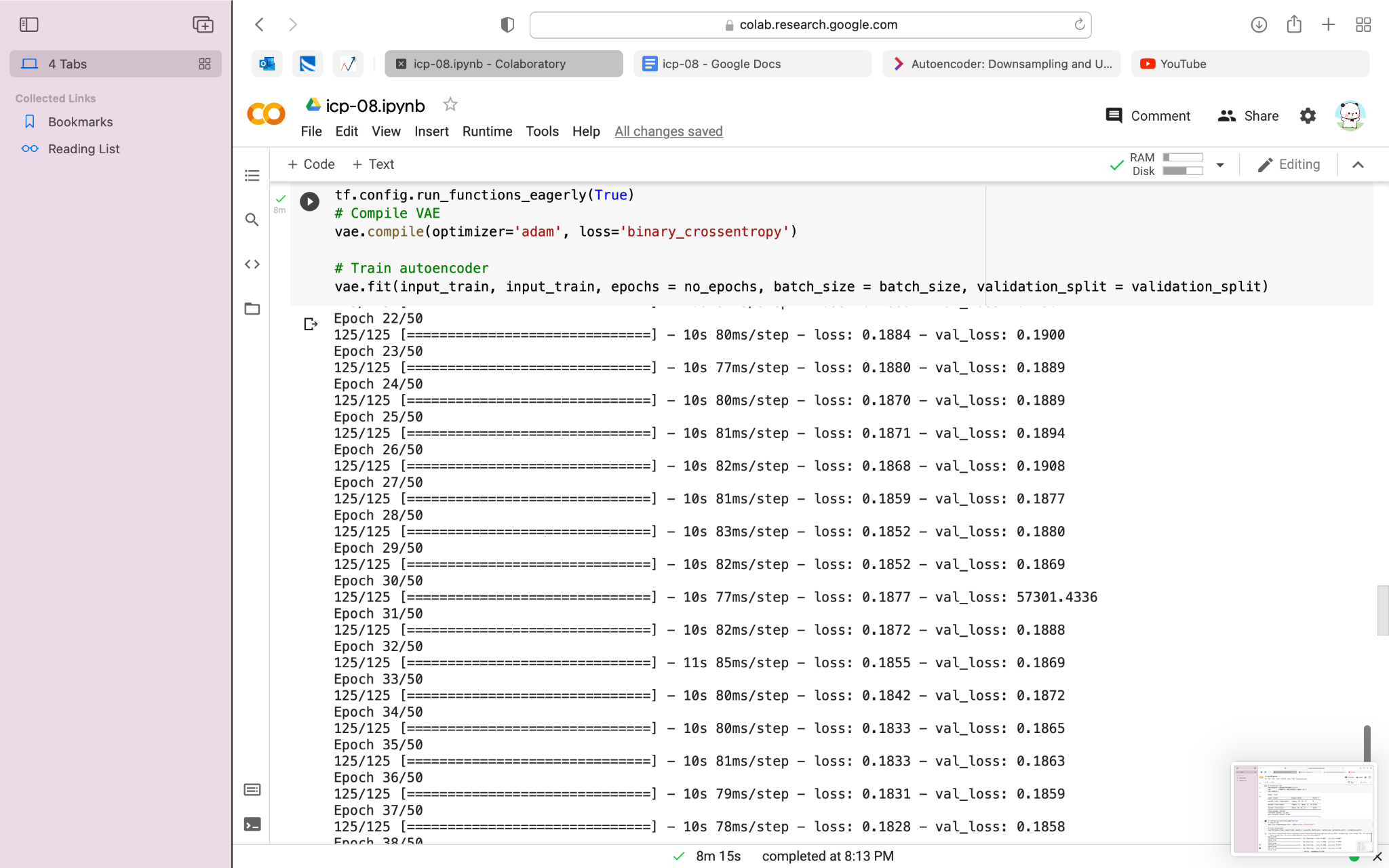
****

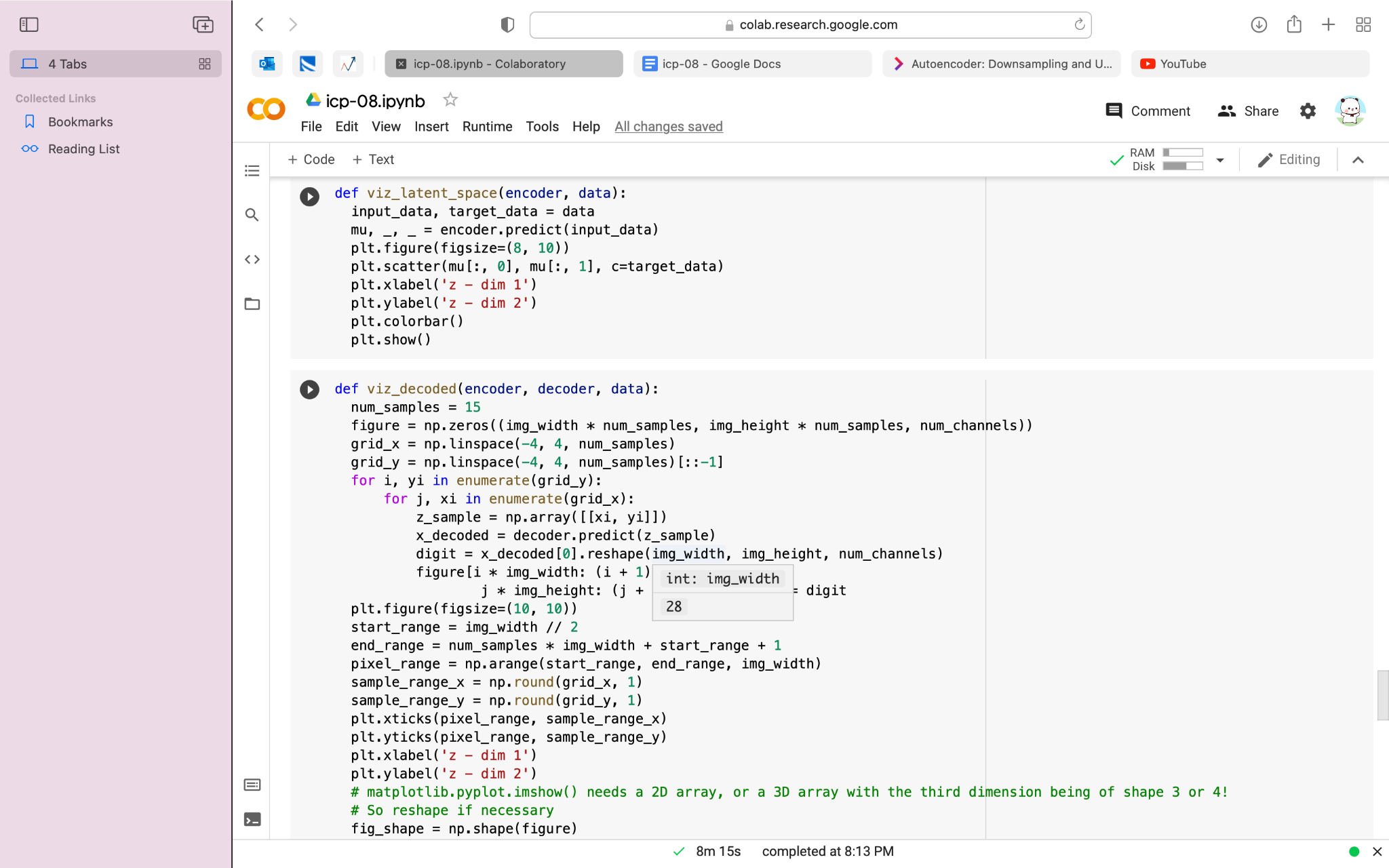
****

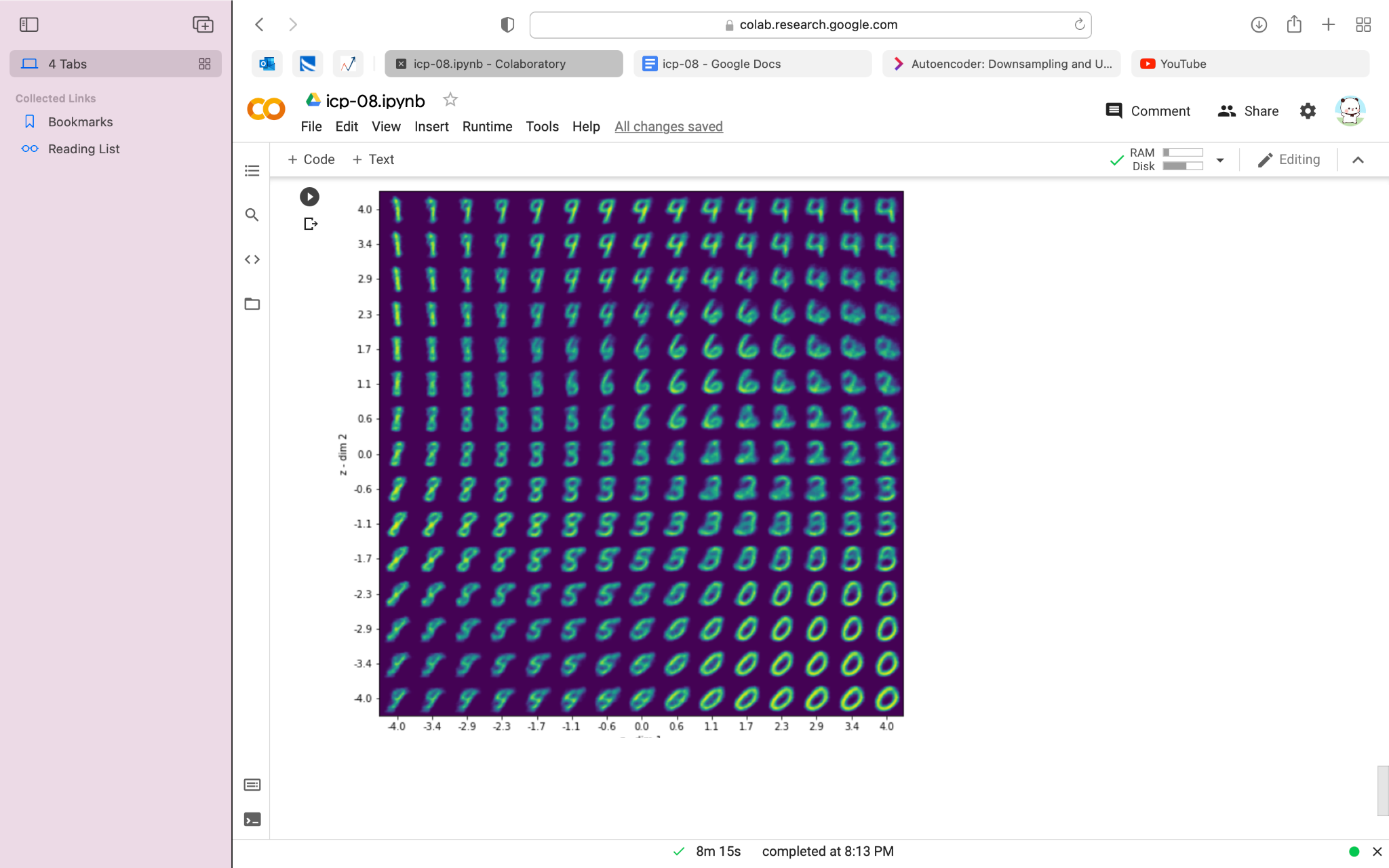
****

****

****

****

****

****

**F)VIDEO LINK:**

[**https://github.com/saidurga-kanuganti/icp-08/blob/main/Mon%20Oct%2025%202021%2010\_49\_47.webm**](https://github.com/saidurga-kanuganti/icp-08/blob/main/Mon%20Oct%2025%202021%2010_49_47.webm)

**G)**The dataset contains different handwritten numbers and have to perform Autoencoder on the given dataset.