<u>Regression Neural Network</u>

By

Charles Silkin

Contents

PROJECT INSTRUCTIONS

Download the image `horse001b.png` from Canvas, containing an image of a horse shape of size 84 × 124 pixels. The goal of this project is to train a regression Neural Network to predict the value of a pixel $I(x, y)$ given its coordinates (x, y). We will use the square loss functions on the training examples $\left(x_i, y_i, I(x_i, y_i)\right), i = 1, \ldots, n$:

$$S(w) = \frac{1}{n}\sum_{i=1}^{n}\left(I(x_i, y_i) - f_w(x_i, y_i)\right)^2$$

where $(x_i, y_i) \in \{1, \ldots, 84\} \times \{1, \ldots, 128\}$ are all $n = 128 \cdot 84 = 10752$ pixels of the image.

All NNs $f_w(x, y)$ we will train have a 2D input (x, y) and a 1D output, but we will experiment with different numbers of hidden layers.

Try to use a GPU and CUDA for faster training. If you want, for better convergence, you could try to standardize the inputs (x_i, y_i) to have zero means and std 1, and the outputs to have zero mean.

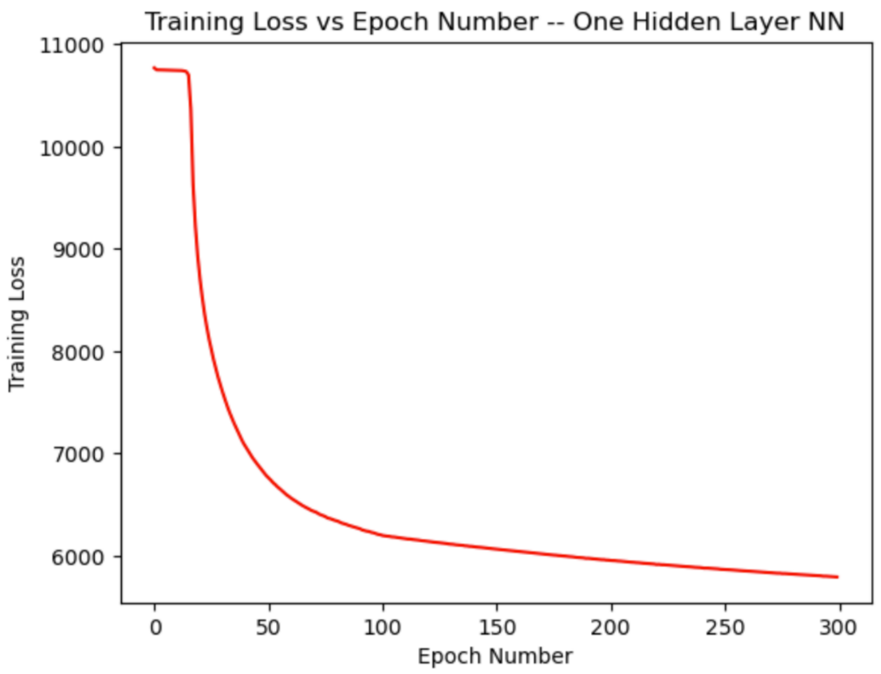
a) Train a NN with one hidden layer containing 128 neurons, followed by ReLU. Train the NN for 300 epochs using the square loss $S(w)$. Use the SGD optimizer with minibatch size 64, and an appropriate learning rate (e.g. 0.003). Reduce the learning rate to half every 100 epochs. Show a plot of the loss function vs epoch number. Display the image reconstructed from the trained NN $f_w(i, j), i \in \{1, \ldots, 84\}, j \in \{1, \ldots, 128\}$.

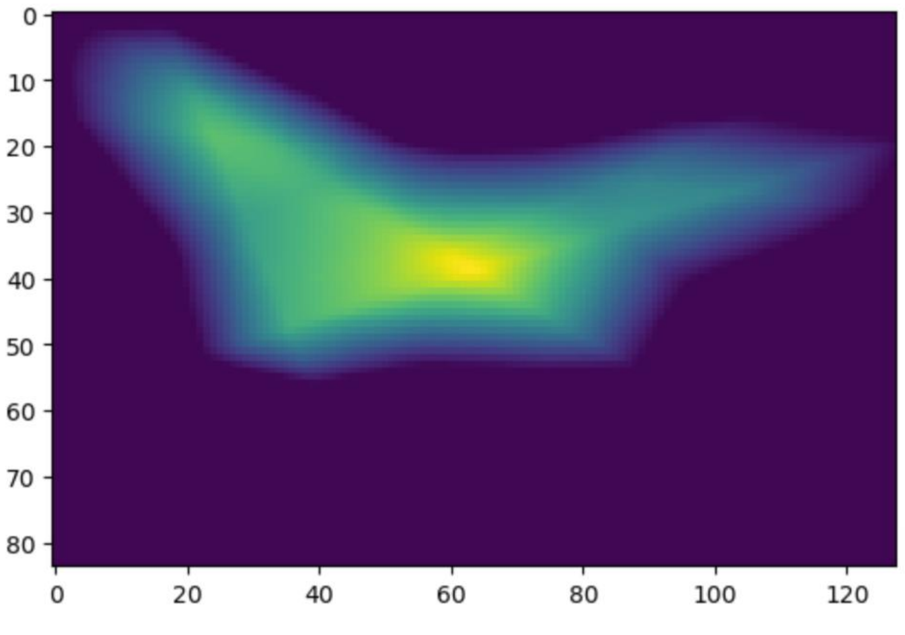
b) Repeat point a) with a NN with two hidden layers, first one with 32 neurons and second one with 128 neurons, each followed by ReLU.

c) Repeat point a) with a NN with three hidden layers, with 32, 64 and 128 neurons respectively, each followed by ReLU.

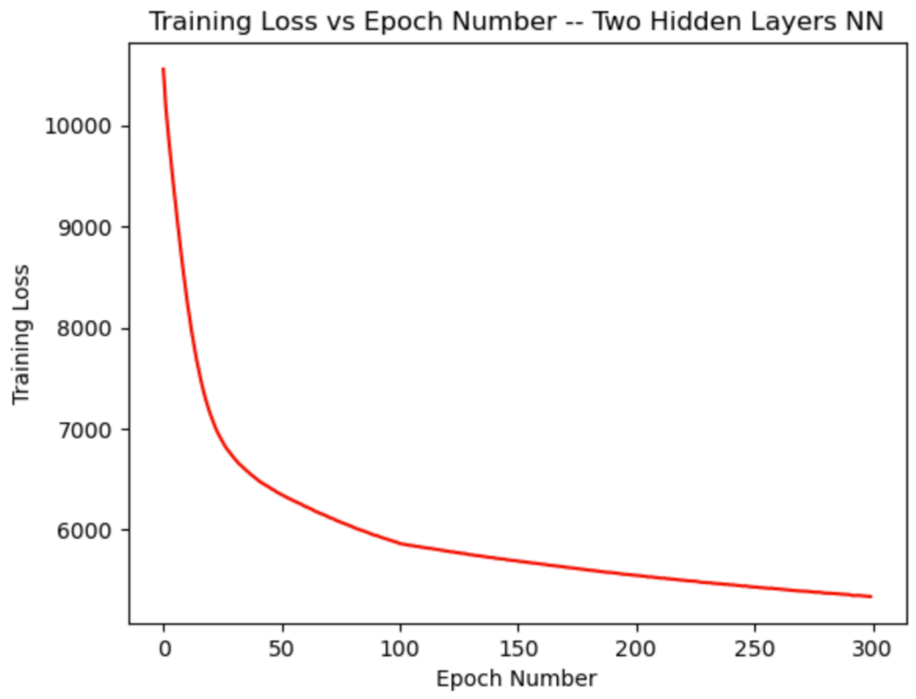
d) Repeat point a) with a NN with four hidden layers, with 32, 64, 128 and 128 neurons respectively, each followed by ReLU. (3 points)

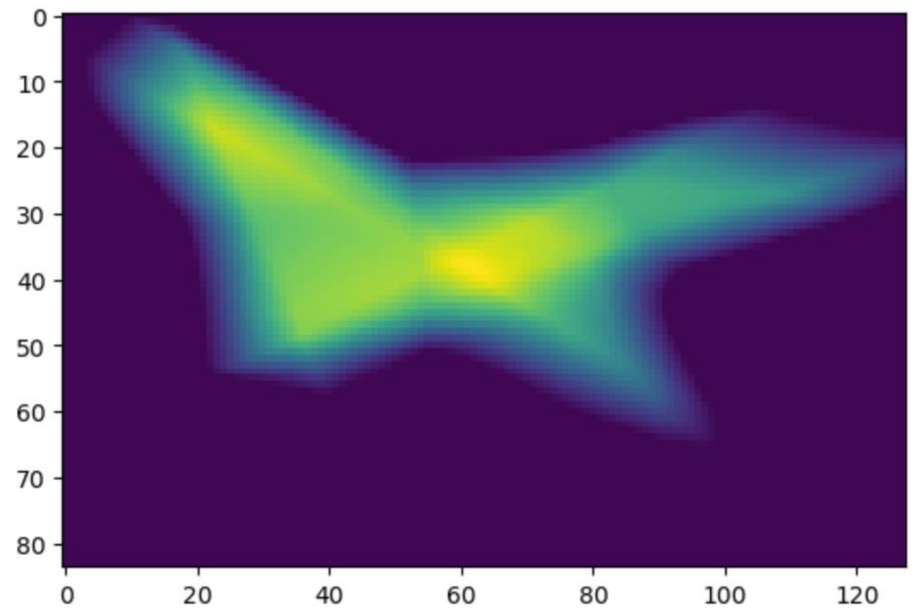
PART A (NN WITH ONE HIDDEN LAYER)



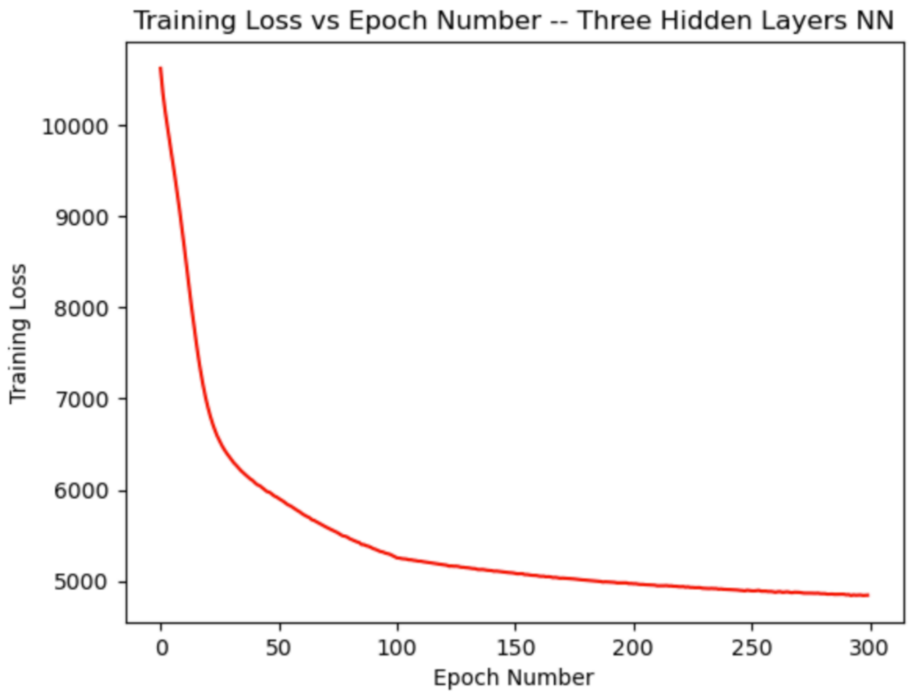


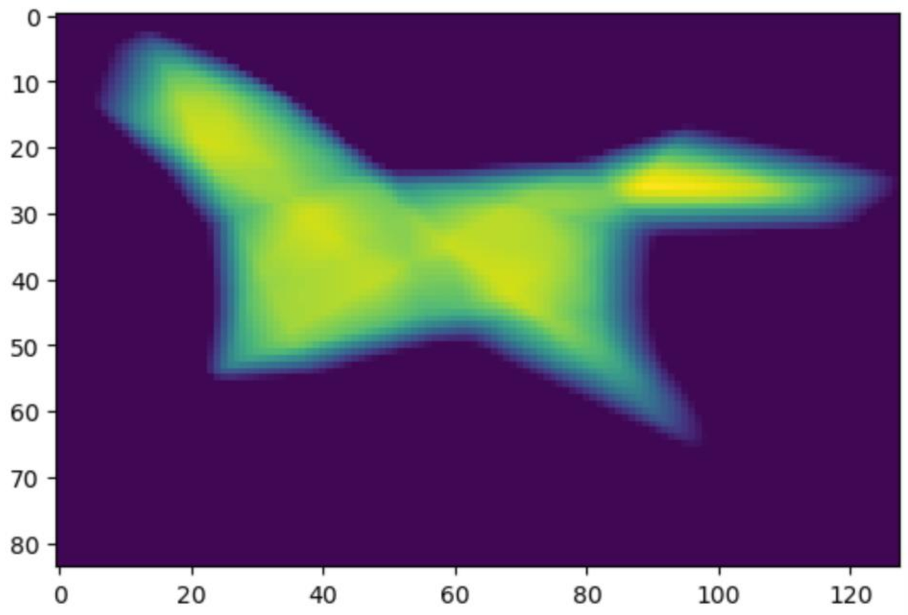
PART B (NN WITH TWO HIDDEN LAYERS)



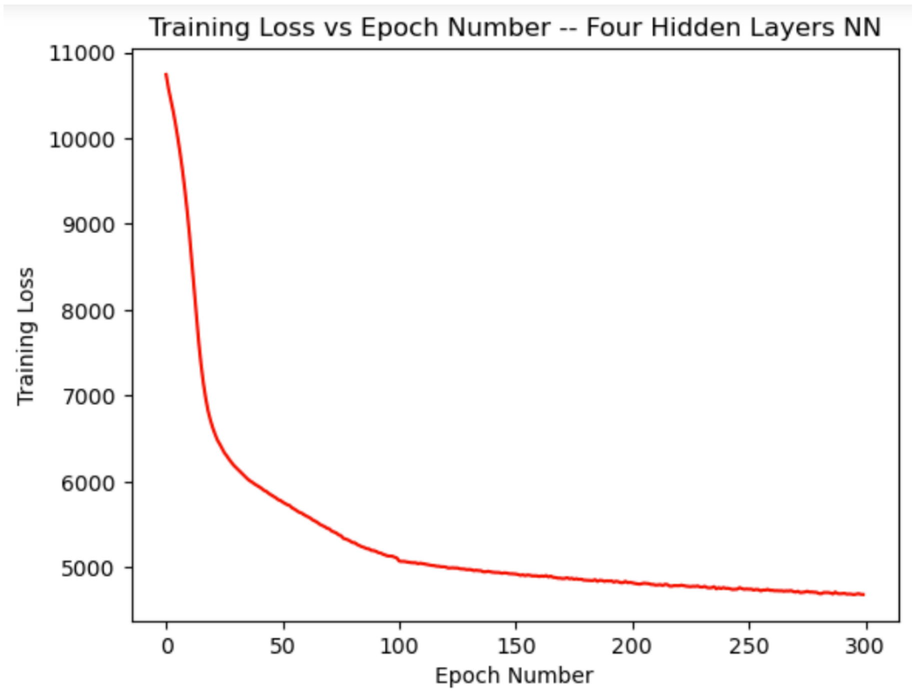


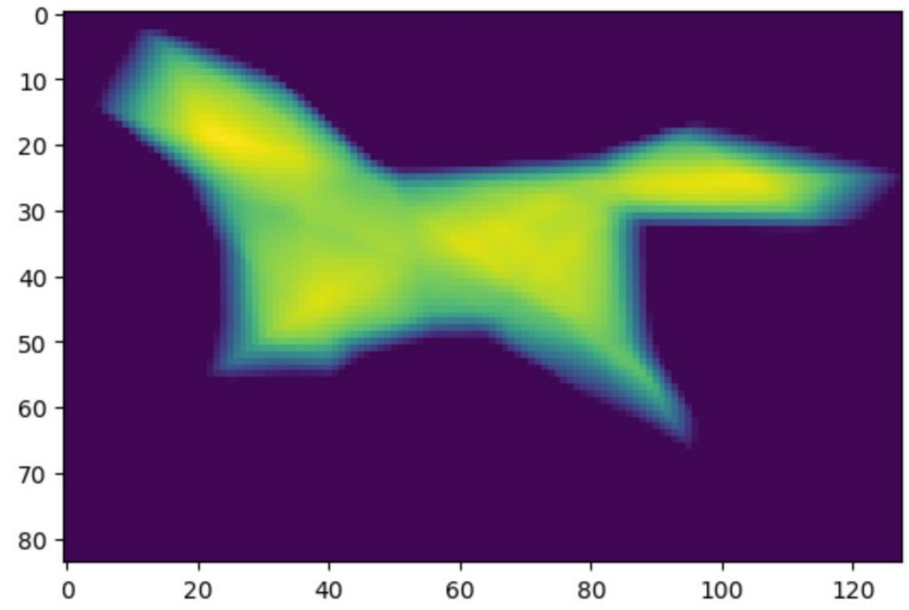
PART C (NN WITH THREE HIDDEN LAYERS)





PART D (NN WITH FOUR HIDDEN LAYERS)





REFERENCES

1. https://towardsdatascience.com/building-a-neural-network-with-a-single-hidden-layer-using-numpy-923be1180dbf
2. https://www.youtube.com/watch?v=ixathu7U-LQ
3. https://www.youtube.com/watch?v=VZyTt1FvmfU
4. https://www.geeksforgeeks.org/how-to-join-tensors-in-pytorch/
5. https://datacarpentry.org/image-processing/aio/index.html
6. https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html
7. https://pytorch.org/docs/stable/generated/torch.nn.Linear.html
8. https://pytorch.org/tutorials/
9. https://towardsdatascience.com/building-neural-network-using-pytorch-84f6e75f9a
10. https://note.nkmk.me/en/python-list-ndarray-1d-to-2d/
11. https://pytorch.org/docs/stable/generated/torch.Tensor.numpy.html