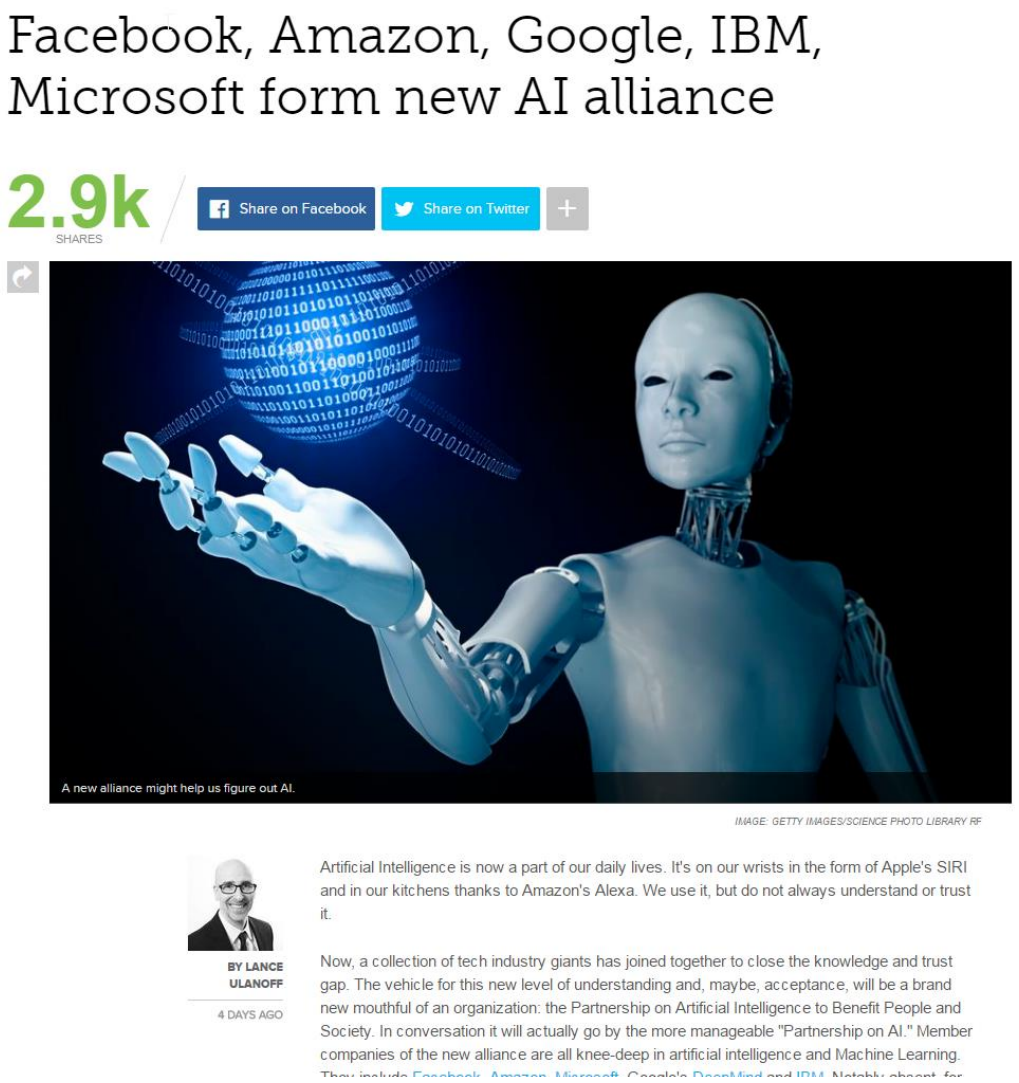
Machine Learning applied to Planetary Sciences

PTYS 595B/495B

Leon Palafox

https://leonpalafox.github.io/MLClass/



Facebook, Amazon, Google, IBM, Microsoft form new AI alliance

A new alliance might help us figure out AI.

IMAGE: GETTY IMAGES/SCIENCE PHOTO LIBRARY RF

BY LANCE ULANOFF

4 DAYS AGO

Artificial Intelligence is now a part of our daily lives. It's on our wrists in the form of Apple's SIRI and in our kitchens thanks to Amazon's Alexa. We use it, but do not always understand or trust it.

Now, a collection of tech industry giants has joined together to close the knowledge and trust gap. The vehicle for this new level of understanding and, maybe, acceptance, will be a brand new mouthful of an organization: the Partnership on Artificial Intelligence to Benefit People and Society. In conversation it will actually go by the more manageable "Partnership on AI." Member companies of the new alliance are all knee-deep in artificial intelligence and Machine Learning. They include Facebook, Amazon, Microsoft, Google's DeepMind and IBM. Notably absent, for

http://mashable.com/2016/09/29/partnership-on-ai/#6V52AXp.Naq8

Rule of thumb

- In general is a bad idea to just use a range between 0 and 1.
  - Since there are many parameters, it can take a long time if we use a random initialization.
- For training, a good initialization range is:

$$\left[-\sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}+1}}, \sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}+1}}\right]$$

Problems of NNs

- We need to answer two questions:
    - How many layers are enough to solve a problem?
    - How many hidden units should we use per layer?
- As you can imagine, training complexity increases as we increase hidden units.
    - This can be reduced by avoiding a full interconnection.
- The elephant in the room is called “Vanishing Gradient”

Vanishing Gradient

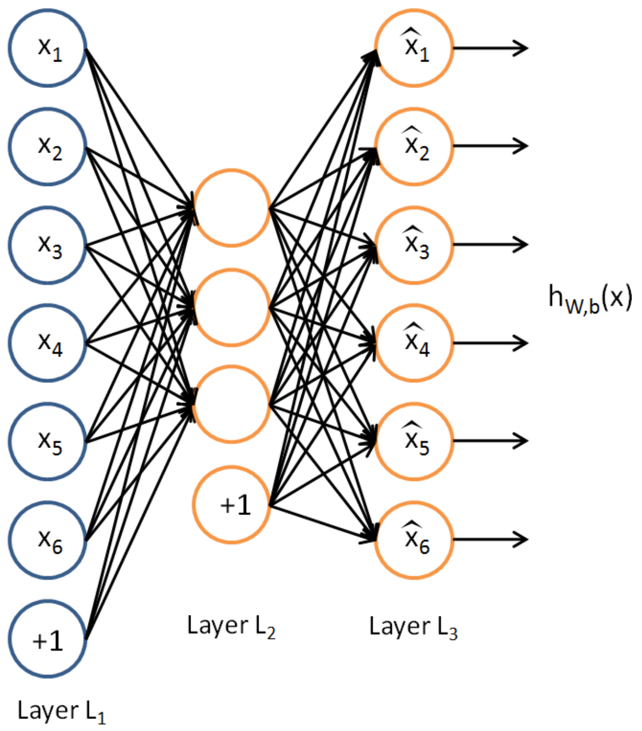
- A problem of NNs, is that our small δs, will become even smaller as we go back in our layers.
- If we have many layers, we are going to end up with really small gradients.
- This will show as negligible updates in the gradient descent equations.
- For many years, before 2006, this was the main reason few people used classic NNs.
  - What is the point of having the power that comes from many layers, if we cannot train it properly anyways?
  - The solution: Pre-training of the individual layers.

The Autoencoder

- The autoencoder is one of many architectures of NNs.
- In the autoencoder we do not use the labels in the dataset.
- Is an unsupervised learning algorithm.
- We do not run things like testing and training datasets.

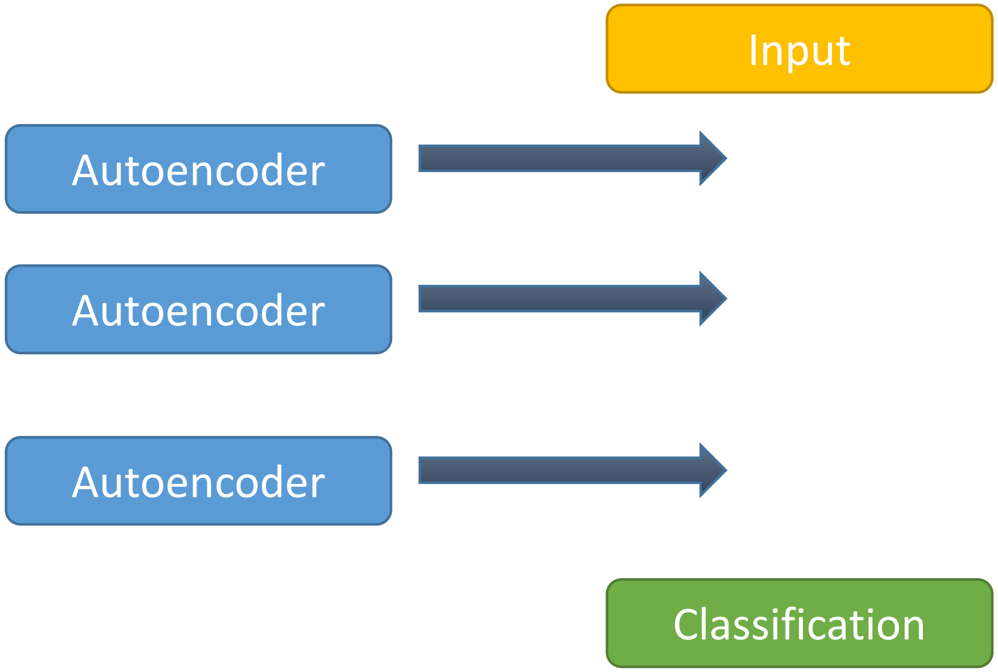
Autoencoders

- An autoencoder is a NN where the output and the input are the same.



Deep Nets

- They are supposed to be deep so:



Activation functions

- The most common activation functions are:
  - Sigmoid (In Wikipedia is called logistic)

$$f(x) = \frac{1}{1+e^{-x}}$$

  - Tanh (we already saw is pretty effective

$$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

  - Rectifier

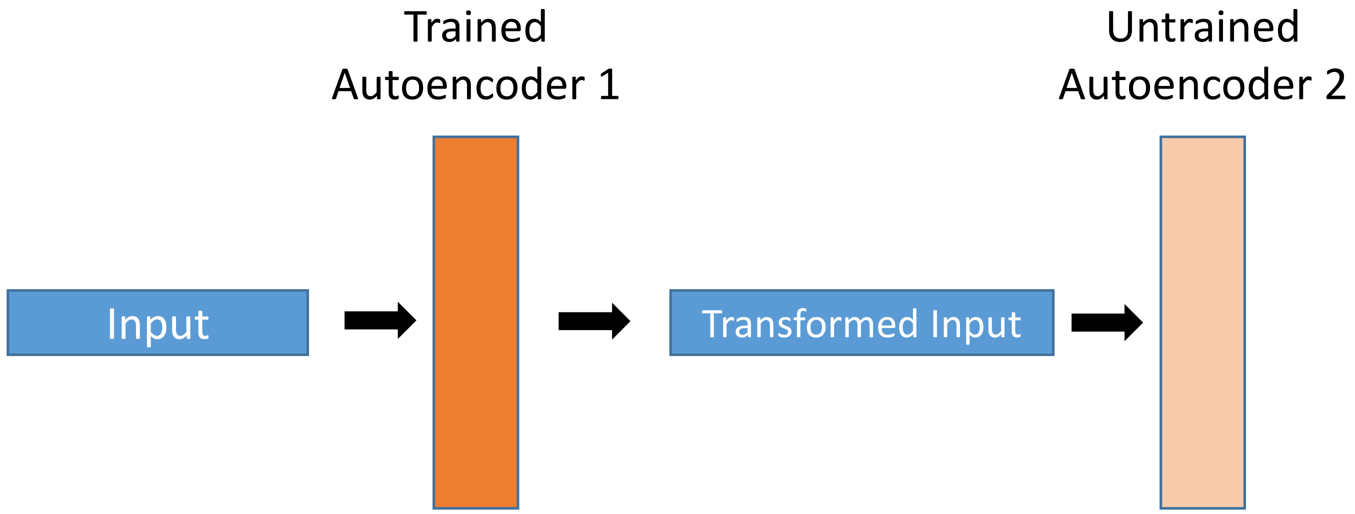
$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$$

Pseudocode

- Step 1: Get your dataset
- Step 2: Design your deep net
    - Hidden Layers (e.g. 2)
    - Activation Functions
- Step 3: Train First autoencoder
    - Using the dataset train the autoencoder with the dataset.
- Step 4: Plug the autoencoder back in the network
    - Substitute the weights in the first layer for those obtained using the autoencoder.

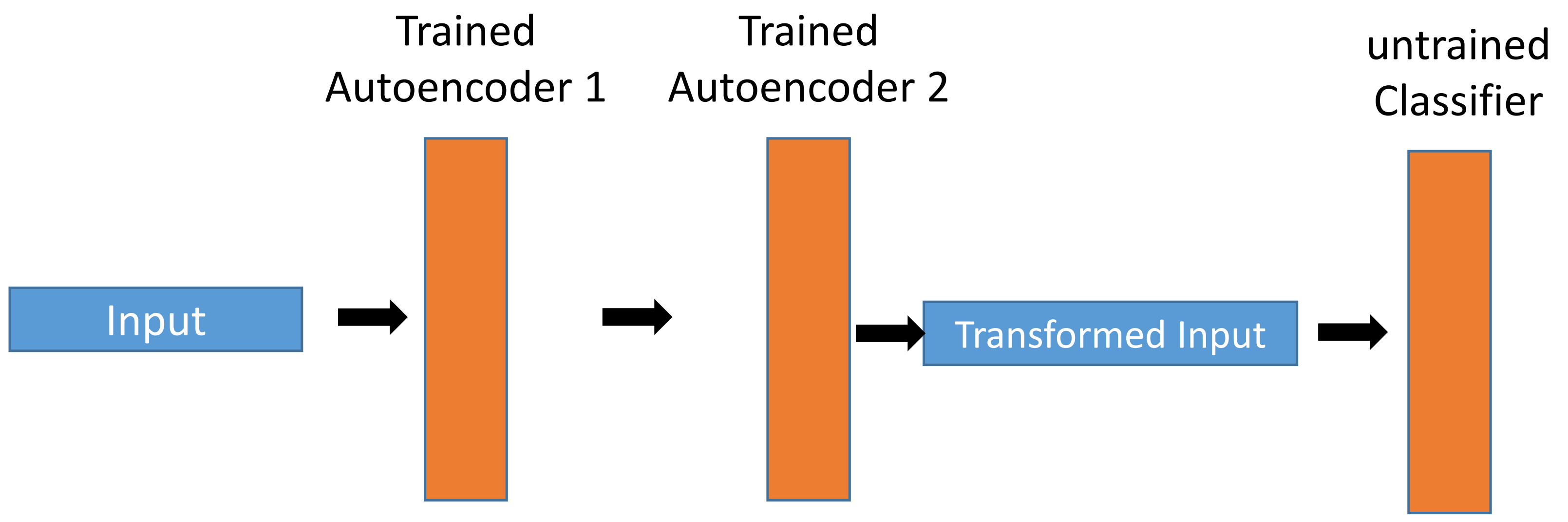
Pseudocode

- Step 5: Transform the dataset using the first layer of the autoencoder, to train the second layer.



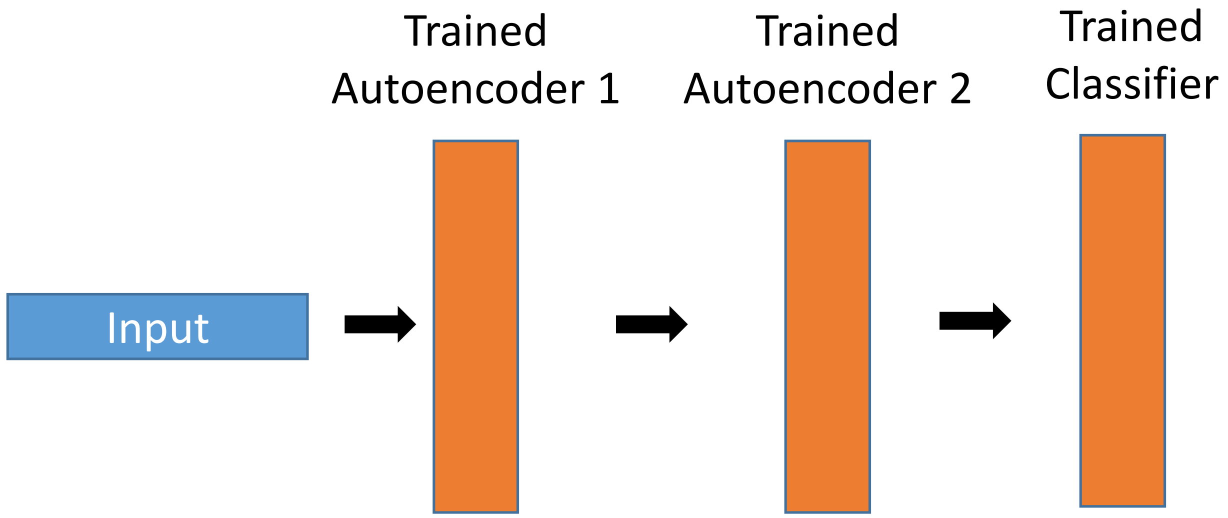
Last layer

- The last layer of a deep net is just a standard classifier.
    - For MNIST we use a softmax classifier (Logistic Regression with multiple classes)



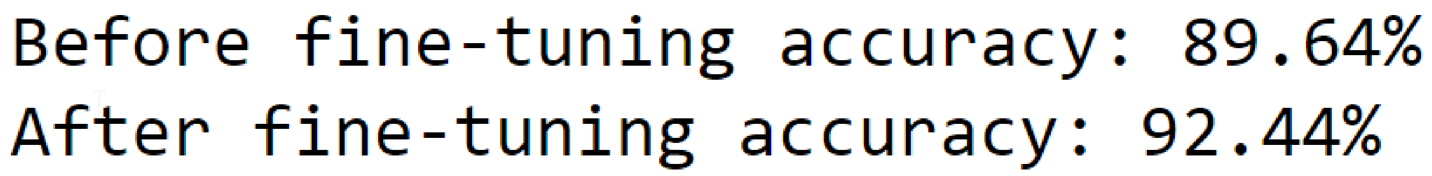
Pseudocode

- Step 6: Train the classifier with the transformed input from the Autoencoder 2
- Step 7: Fine tune the whole architecture (400 iteration should be fine)



Accuracy

- Before fine tuning the accuracy is already very good



```
Before fine-tuning accuracy: 89.64%
After fine-tuning accuracy: 92.44%
```