Intro to Deep Learning

Summary

This workshop will introduce the conceptual basis of deep learning with neural networks and introduce the Keras library workflow.

What is Deep Learning?

In a nutshell, it's applied linear algebra and calculus running on the latest hardware. Yay, math!

The "deep" refers to the number of layers of data transformation used to sift for meaningful representations. Imagine a chain of logisic regressions feeding into eachother.

Deep learning is the application of many different representations of data to identify meaningful relationships in data. Explore high dimensional representations of the data.

The key component of deep learning sending representation results through an optimizer that updates weights and runs the inputs with the updated weights.

Neural Networks are not new. Their astounding results are!

Pitts and McCulloch created the first computational model for neural networks in 1943. This was a computational modle of how biological neurons fire. In 1958, Rosenblatt created the first perceptron, which is a binary classifier. Perceptrons create linear boundries between datapoints in spaces. Ivakhnenko Ivakhnenko and Lapa created the first network with multiple layers in 1965.

What Deep Learning is *not*

Cognition or thinking or inaccessible mathematics. Nor is it generalized.

Always better or always worse than classical machine learning. **It depends** is the answer.

Free from bias and prejudice. (Biased inputs mean biased outputs)

Garbage in = Garbage out and bias in = bias out

As an example, researchers building criminality prediction algorithm trained a neural network with labeled pictures. The criminal pictures were mug shots from convicts. For the "non-criminal" images, researchers used a set of royalty-free images of fashion models. Imagine the results! The more someone looks like a model, the less likely the criminality!

Incentivized, ignorant, and malicious actors will claim "the data doesn't lie". They wash their hands claiming "science" and "computer algorithms". They have no incentive to question assumptions.

Takeaway: *many* problems need the model to be interpretable.

It's our ethical duty to shine the light, to work to make things better not worse.

We can't put a neural network *on the stand* to explain itself and be cross-examined.

Neural networks are fundamentally black box functions.

Why and when to use ANNs vs. Classical ML tools

When we have rich, perceptual data such as text, images, audio, or video

When we want automated feature engineering, but remember the curse of dimensionality!

When we have the resources and time

When interpretability is not a critical issue.

Why and when to use Classical ML over ANNs

- Easier to interpret. We can put a decision tree on the stand. It'll stand up in court.
- Works better on small data
- Financially and computationally cheap

Proper Feeding of Neural Nets

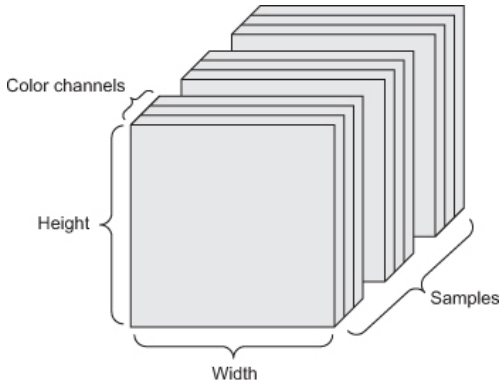
We need to feed tensors to neural nets. Tensors are n-dimensional matrices of scalars.

Vector data is a 2-dimensionsal tensor (1 dimension for the samples, **features**)

Time series data or sequence data are 3D tensors of shape (samples, time-steps, features)

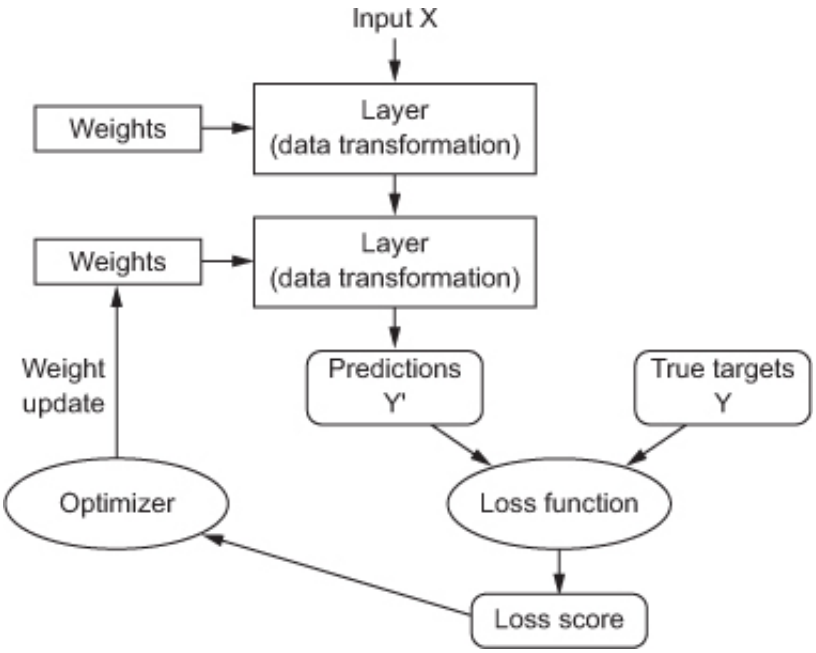
Images—4D tensors of shape(samples,height,width,channels)or(samples, channels, height, width)

Video —5D tensors of shape (samples, frames, height, width, color channels)

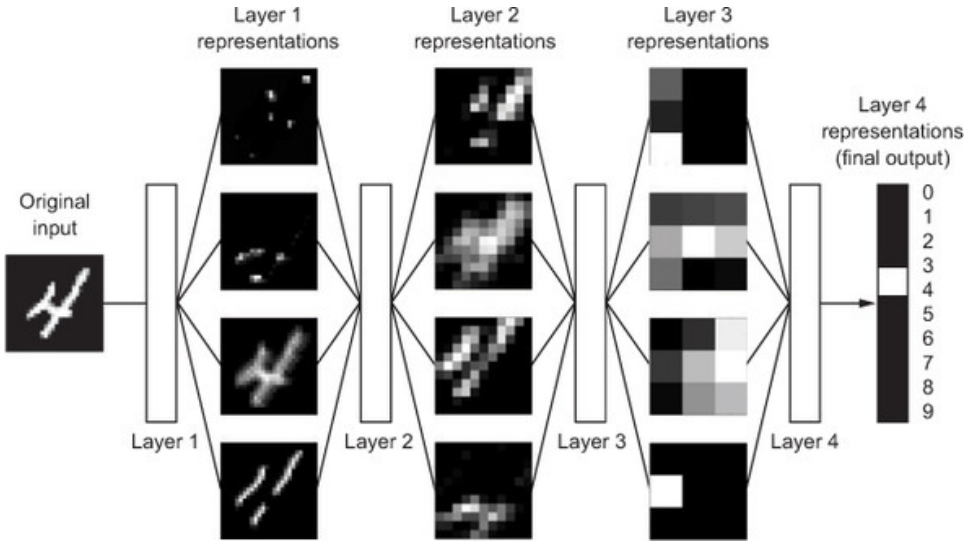


It's important to note that the inputs need to all have the same shape and must be in a tensor format. This means that we generally have to do more pre-processing. We can't send images straight into each layer. We need to convert all the images to tensors.

Anatomy of a Deep Learning Network



- In deep learning, we perform many data representations and transformations to compare predictions vs. true targets. The accuracy of the prediction is fed into an optimizer, and the optimizer updates the weights on different layers or features. Then, we run the data through the same layers with updated weights.
- This is the "learning" process: the feeding back of the result of predictions (backpropogation) through an optimization function that updates weights on layers that run on the data again, making for more effective predictions.
- Each layer is the application of many geometric transformations of the tensor data

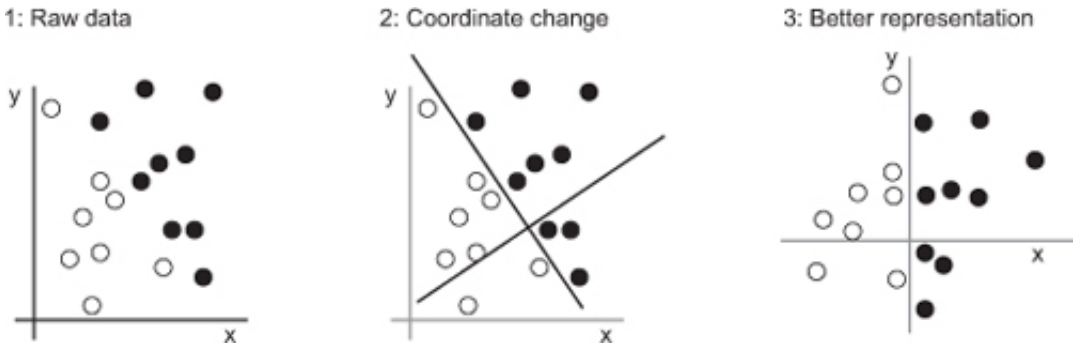


What is each layer actually doing?

Geometric transformations in high dimensional space. "Uncrumpling paper balls is what machine learning is about: finding neat representations for complex, highly folded data manifolds. Each layer in a deep network applies a transformation that disentangles the data a little" - Chollet

This figure shows the original data and a new representation. We obtain new representations of the data by changing our axes to fit the story the data tells. Here we get a new representation of the data in such a way that we can describe data with a simple rule:

> “Black points are such that x > 0,” or “White points are such that x < 0.”



Backpropogation is how the network "learns"

With our `optimizer`, training happens within what’s called a *training loop*, which works as follows. Repeat these steps in a loop, as long as necessary:

1. Draw a batch of training samples `x` and corresponding targets `y`.
2. Run the network on `x` to obtain predictions `y_pred`.
3. Compute the loss of the network on the batch, a measure of the mismatch between `y_pred` and `y`.
4. Compute the gradient of the loss with regard to the network’s parameters (a *backward pass*).
5. Move the parameters a little in the opposite direction from the gradient—for example `W = step * gradient`—thus reducing the loss on the batch a bit.

How does the learning actually happen?

Questions

What's the relationsihp betweeen artificial neural networks and real neurons?

What are neural networks and deep learning?

How are artificial neural networks (ANNs) different than other machine learning algorithms?

When should we use ANNs vs. classical machine learning?

Tremendous volumes of unstructured data? An ANN may be the right tool

The central problem in deep learning is to meaningully transform data to learn useful representations of the input data so those representations get us closer to the desired output.x

What's a representation? A different way to look at data. For example, the RGB representation of "red" is [1, 0, 0] not the string "red".

Keras Workflow

0. Prep and load your data. You'll nemeaning it's already a tensor or you've converted your data to be a tensor)
1. Create the model
2. Add layer(s) with .add
3. Compile the model with .compile (to configure the learning parameters)
4. Fit the model with .fit
5. Evaluate model performance with .evaluate
6. Produce predictions on new data with .predict
7. Decode the prediction data if necessary back to its original representation (numbers, text, images)