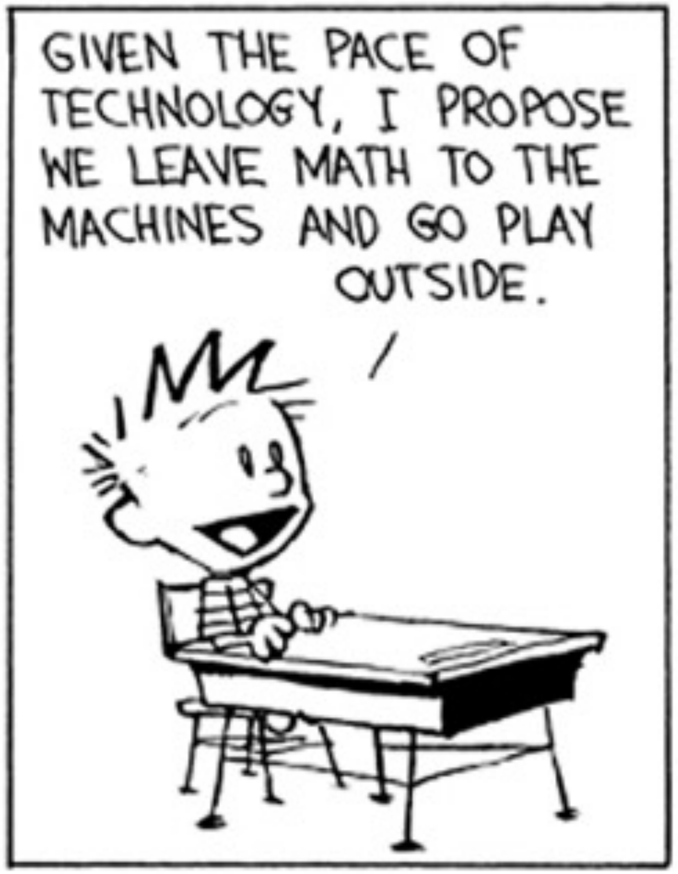
Introduction to Deep Learning

Christina Kumler

Christina.E.Kumler@noaa.gov

CIRES – NOAA GSL


GIVEN THE PACE OF TECHNOLOGY, I PROPOSE WE LEAVE MATH TO THE MACHINES AND GO PLAY OUTSIDE.

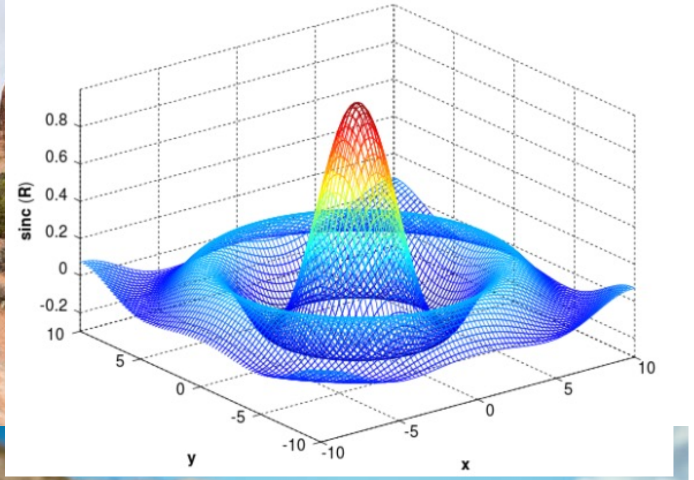
Agenda

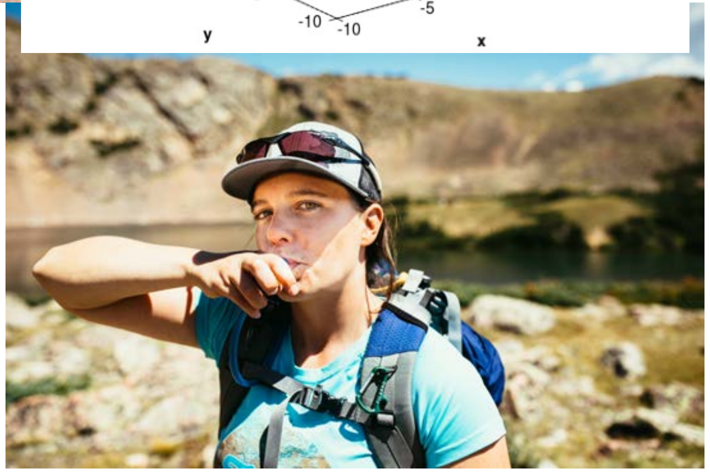
1. Quick introduction to Christina
2. What is Machine Learning?
3. What are Neural Networks?
4. What is Deep Learning?
    1. Deep Learning in Environmental Science
    2. The Infamous U-Net
5. Exercise
6. Additional Resources

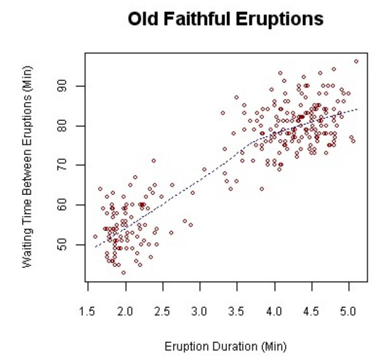
Who is Christina Kumler?

Who is Christina Kumler?

- A math, computer, and weather scientist!
- Hobbies?
  - Hiking, triathlons, cooking, eating, photography, traveling, and video gaming
- Applied Mathematics
  - Major at CU Boulder!
  - The study of math both in theory and application
  - “Modeling”
    - This means seeing something cool in the world and wanting to better understand it and predict its behaviors
      - We can represent all sorts of real-life animals, events, and things using numbers
- Meteorology, atmospheric science, and oceanic science
  - Masters at University of Miami – FL - RSMAS







What is Machine Learning?

Machine learning is a bakery and each goodie within machine learning can be classified into a type (cookie, brownie, cake) and then further into flavors (chocolate, peanut butter, chocolate peanut butter)

What is Machine Learning?

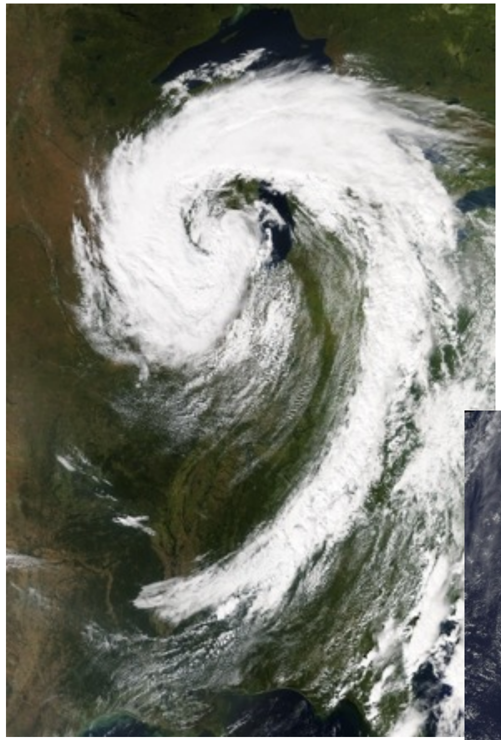
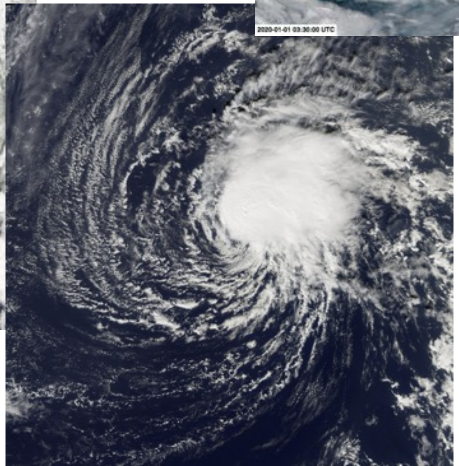
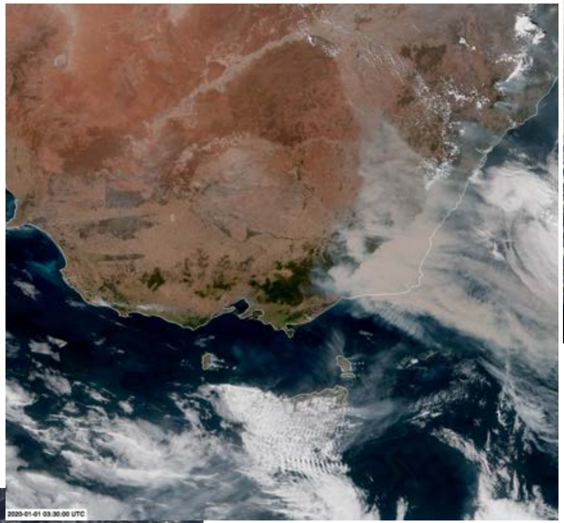
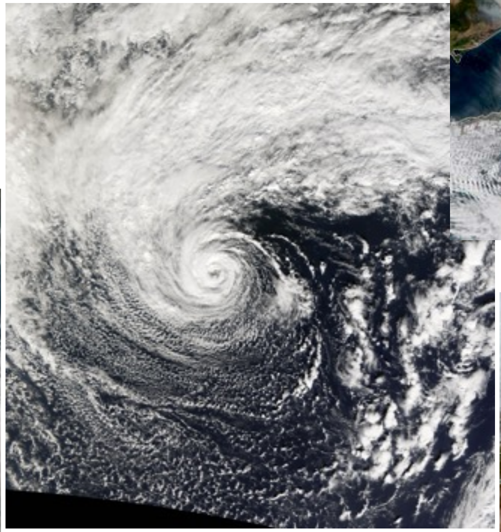
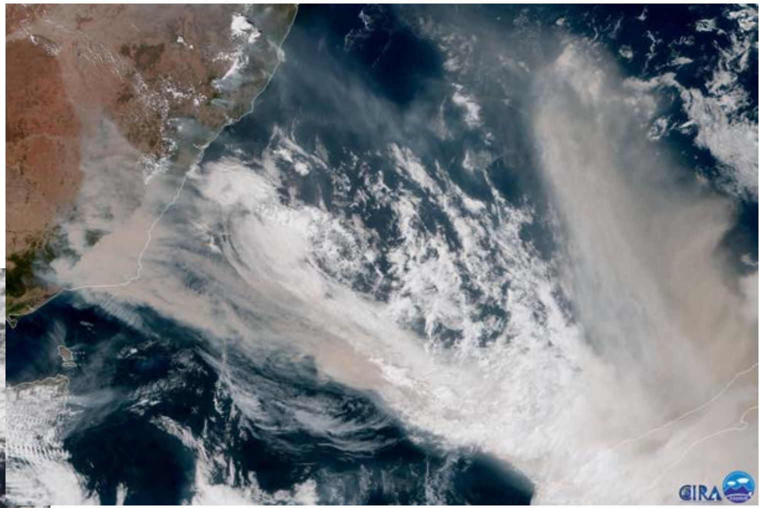
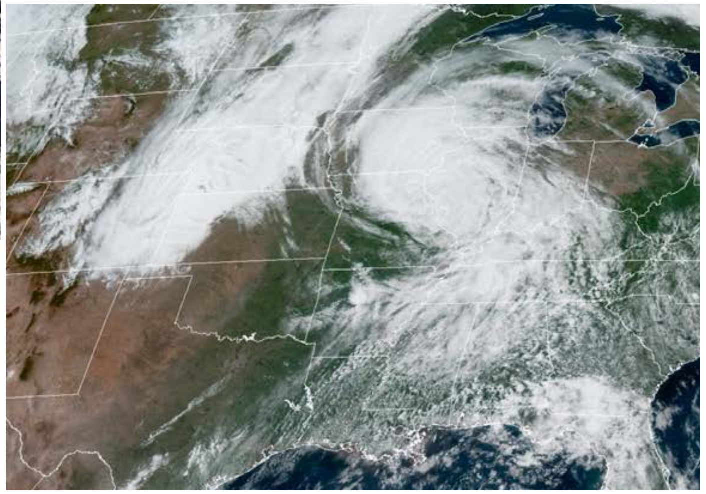
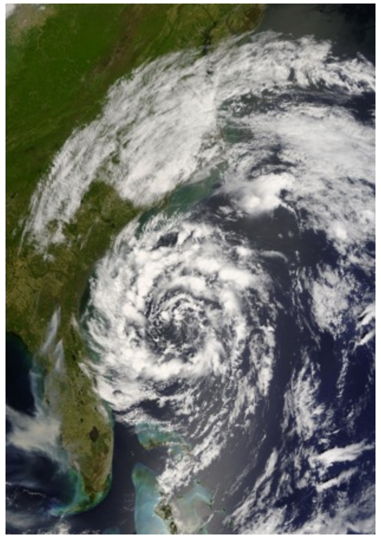
Common Terms:

- Artificial Intelligence (AI)
  - Programs and models that mimic how humans learn new things, complete tasks, and problem solve
- Machine Learning (ML)
  - This is a type of AI
  - Where a computer or model learns from data without being told specific rules or ways to learn.
    - The human might or might not give it the “answers” to what its trying to learn, but it learns how to build a routine to solve the problem itself.

What is Machine Learning?

How do we as people learn?

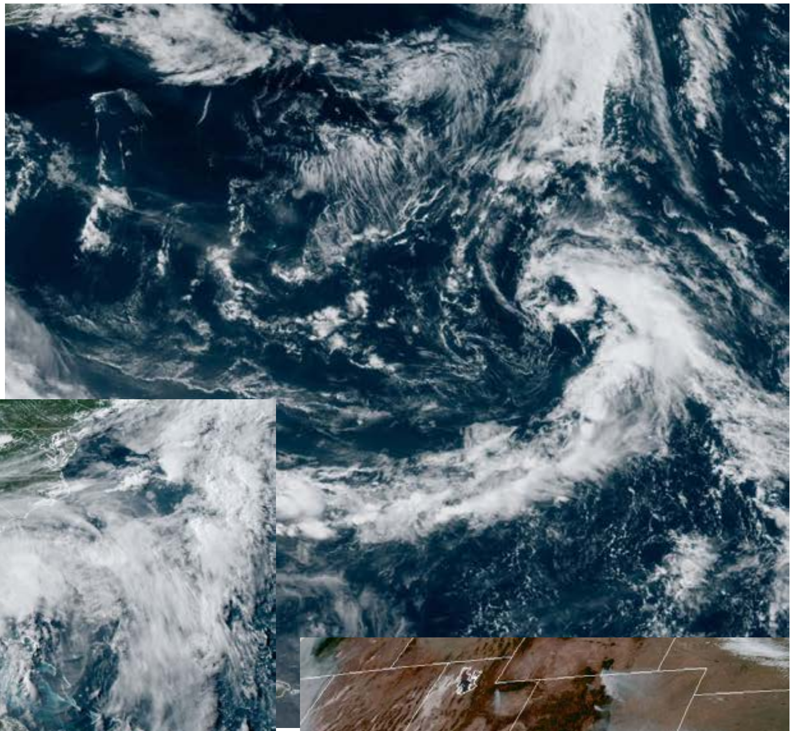
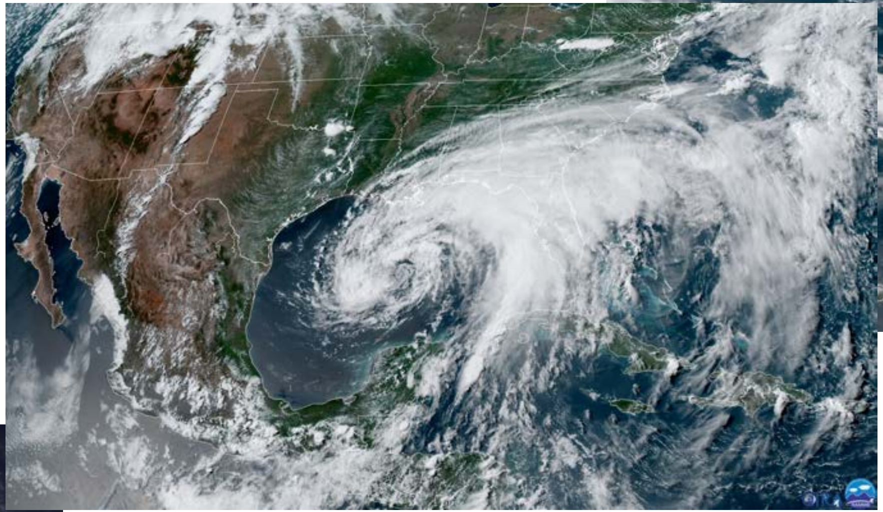
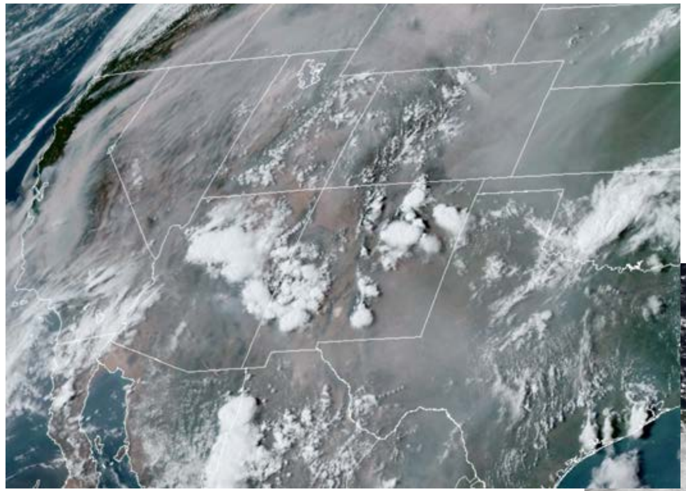
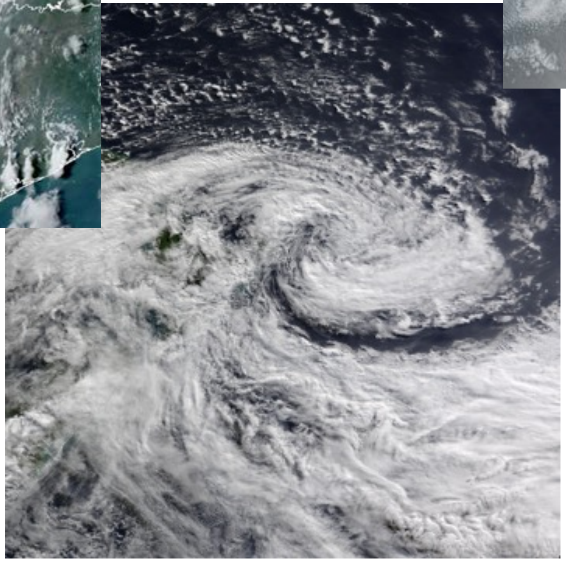
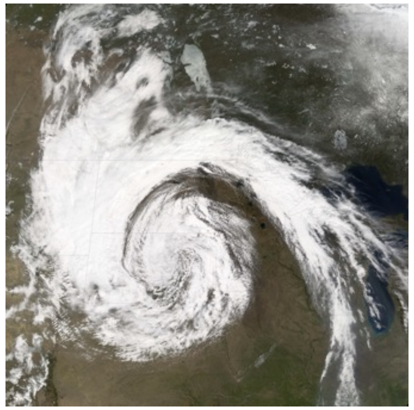
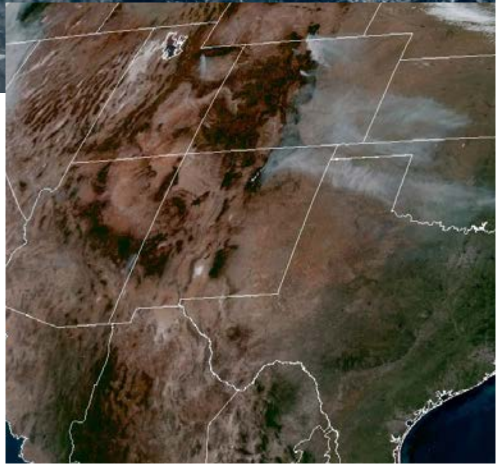
- Unsupervised

What is Machine Learning?

How do we as people learn?

- Supervised

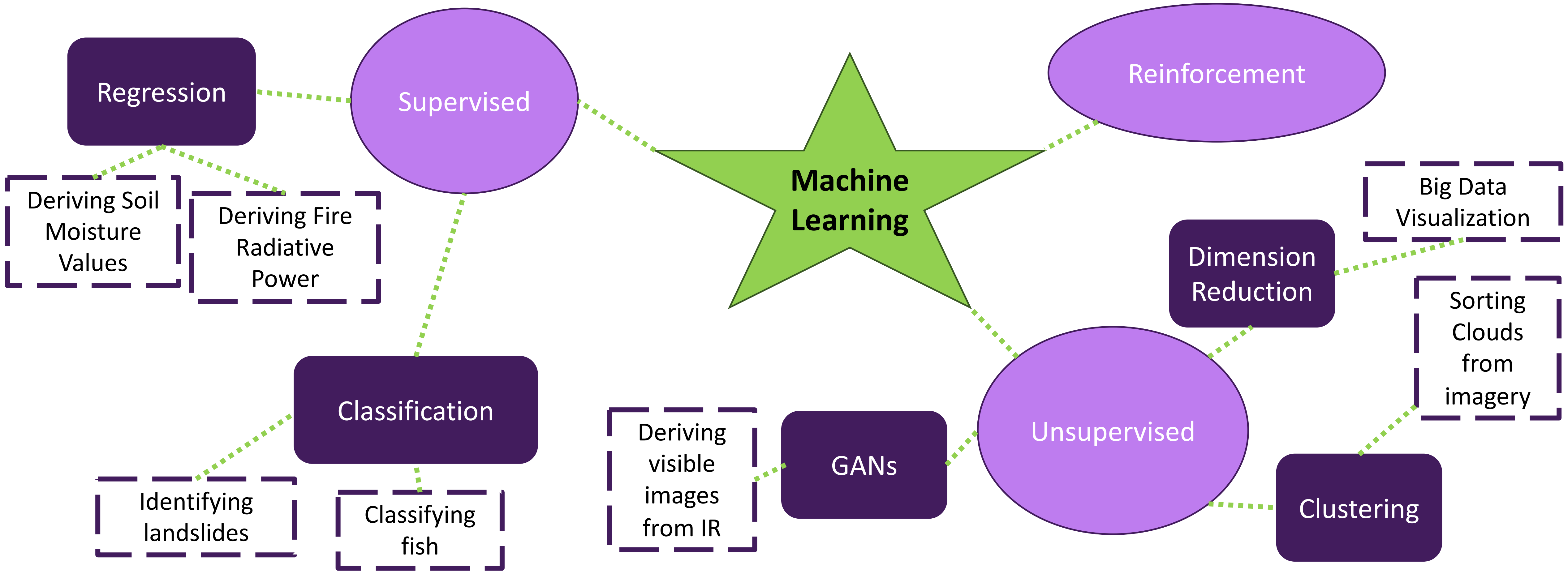
What is Machine Learning?

How do we as people try to find solutions to complex problems?

1. Start with looking for repeat patterns/features
2. Look at general similarities or guidelines to solve problem.
3. Work our way into more complex connections.
4. Assigns weights to these guidelines/rules.
5. Get stumped when something is presented that we haven’t seen before.
6. Repeat steps 1 and 4.

What is Machine Learning?

Problems that can typically be solved by machine learning:



What is Machine Learning?

Pros:

- Most common machine learning models are written in the accessible python language using certain packages such as:
    - Keras
    - Sklearn
- Very easy to obtain packages and set up a "basic machine learning model" quickly
- Once a model is trained, the "inference" stage often runs very quickly
- If trained correctly, they can be remarkably good at their jobs.

Cons:

- Unfortunately, machine learning can be very finicky about environments. Sometimes it takes some time to set up an environment. Sometimes trained models cannot be called in different environments.
- Training deep learning models takes a lot of computational power and time.
    - This can be dramatically increased with access to GPUs.
- Translating machine learning models to non-python languages is a slowly developing field.
- Overfitting.
    - Too little training data.
    - Not properly trained model.

What is Machine Learning?

- As geoscientists, we absolutely cannot just take machine learning model frameworks and throw it at a geoscience problem. There are many unique challenges to our type of problems that must be considered:
  - Imbalanced datasets
    - Oftentimes we're interested in rare-occurring events
  - Time dependency
  - Location Dependency
  - How labels are obtained
    - And what biases these labels will give to the model
  - Many applications require high-trust to ever be implemented
    - Forecasters do not like using "black box" models when situations are life and death
  - Need to follow laws of physics

What is Machine Learning?

- Because of the previous reasons, geoscientists will never be “replaced” by machine learning algorithms. They are critically important in applying their field expertise to the successful development and implementation of models.
  - Our roles are as important as computer scientists and interdisciplinary communication is vital in model development and application.
  - **People solve problems, technology is the tool to do so.** – Andrew Foster, fellow AI2ES member
- Neural networks are magic:
  - They have the potential to solve many problems and quickly, without understanding how the model actually comes up with the solutions to the problem.
- Neural networks are not magic:
  - Cannot throw random problems and datasets at machine learning and expect a well-performing model.

What are Neural Networks?

To understand deep learning, we spend most of our time understanding neural networks and how they are both magical and complicated.

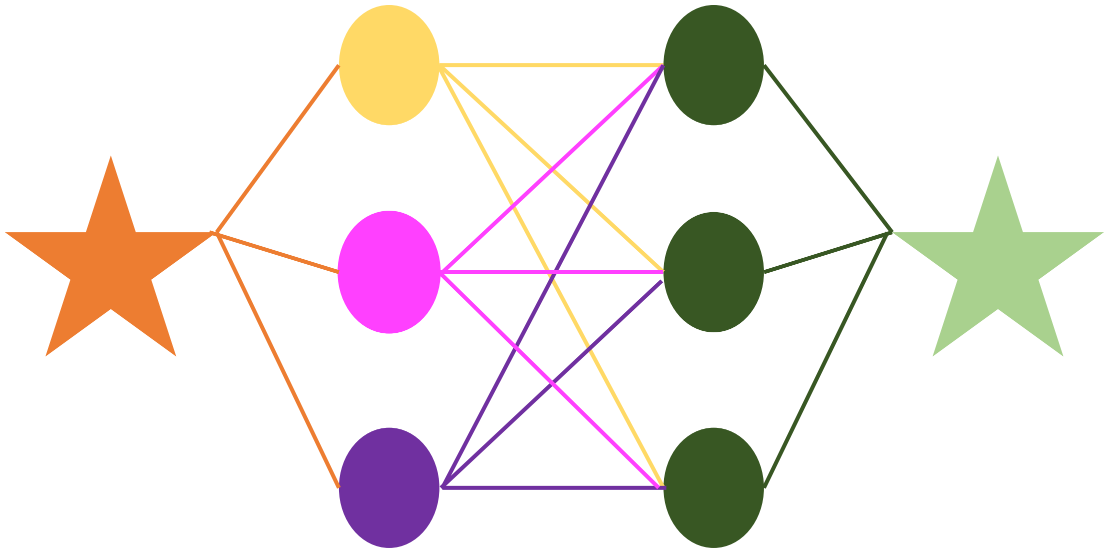
What are Neural Networks?

The ingredients and buzz words of neural networks you should look out for in upcoming slides. When these come up, write them down:

- *Weight*
- *Bias*
- *Activation function*
- *Black Box*
- *Back propagation*
- *Gradient Descent*
- *Loss Function*
- *Hyper Parameters*
- *Epoch*
- *Batch Size*
- *Noise*
- *Dropout*

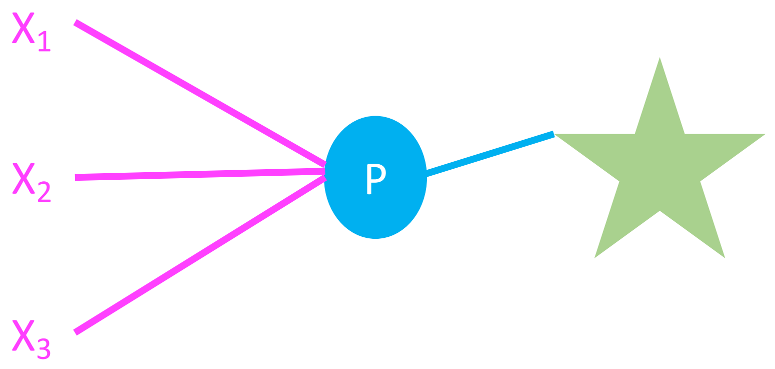
What are Neural Networks?

- Neural Network
  - Human brains, your brains, have “neurons” inside that fire up every time they are called to action!
  - The way we set up the computer program works very similarly. The computer has an “input” that then follows to connected steps, and then connects to more steps, and then “converges” or comes back to one answer



What are Neural Networks?

Perceptron neurons were developed in the 1950's and 1960's attributed to Frank Rosenblatt but also designed earlier by Warren McCulloch and Walter Pitts.[1]

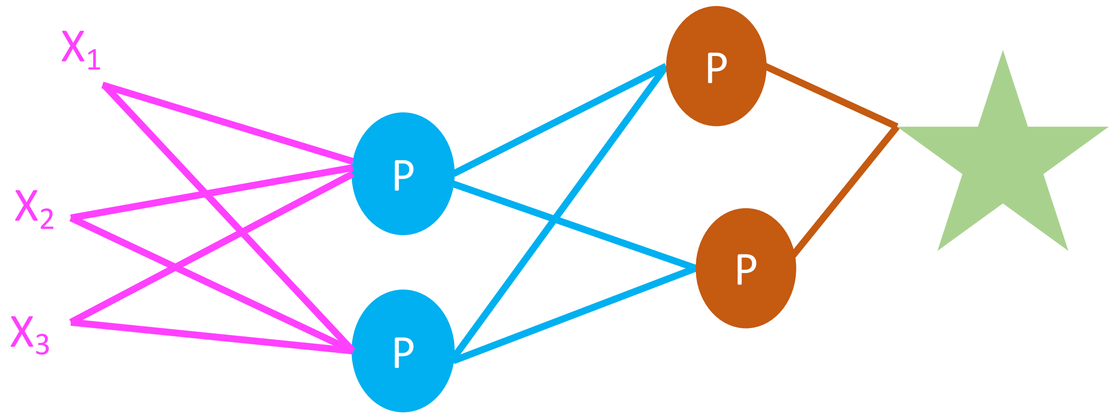


Weight: a parameter being optimized by the model. $W_n * X_n \rightarrow P$

A perceptron produces a single output from neuron inputs, X_n, that are weighted heading into the perceptron layer to make the best "decision" or "guess" to get to an output 0 or 1.

What are Neural Networks?

Now, we can have two layers of perceptrons:

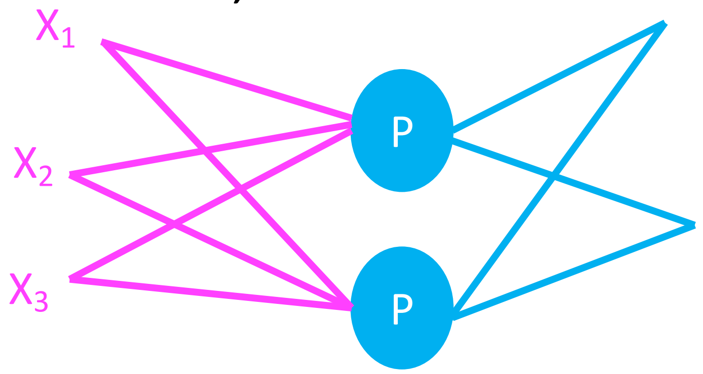


Note: The perceptrons are still only producing one output, but this output is called in the second layer by multiple things in the model.

That first layer still calculates weights from the inputs, but now there's a second layer makes decisions based off the first layers weights, and suddenly this layer is making slightly more complex and abstract decisions to get to a desired output.

What are Neural Networks?

An *activation function* is what prepares the output from the perceptron. This step is where non-linearity is introduced, if a non-linear function is selected.



The *bias* is a constant value inputted to decrease variance and make a neural network more flexible and less likely to overfit the dataset.

First, the weights are applied to each input:

$W_1 * X_1, W_2 * X_2, \ldots, W_n * X_n$

Then, the training calculates a weighted sum:

$W_1 * X_1 + W_2 * X_2 + \ldots + W_n * X_n$

Lastly, a bias, b, is added to the weighted sum and prepared to be given to the activation function:

Activation function$(W_1 * X_1 + W_2 * X_2 + \ldots + W_n * X_n + b)$

What are Neural Networks?

Final thoughts on the basic neuron concept:

- We can think about activation functions as the translators between the weighted-biased sum of one neuron input into another.
- You might hear “sigmoid neuron” which works the same way but the input and outputs can span the entire range of [0,1].
  - Better at handling small changes to the output of a neuron because the input weights and bias can handle a range.

What makes neural networks seem magical?

- The model derives the weights automatically on its own. We don’t tell it what to do or why.
  - We don’t fully understand why the model does what it does and this has lead neural networks to be referred to with the term “*black box*.”

What are Neural Networks?

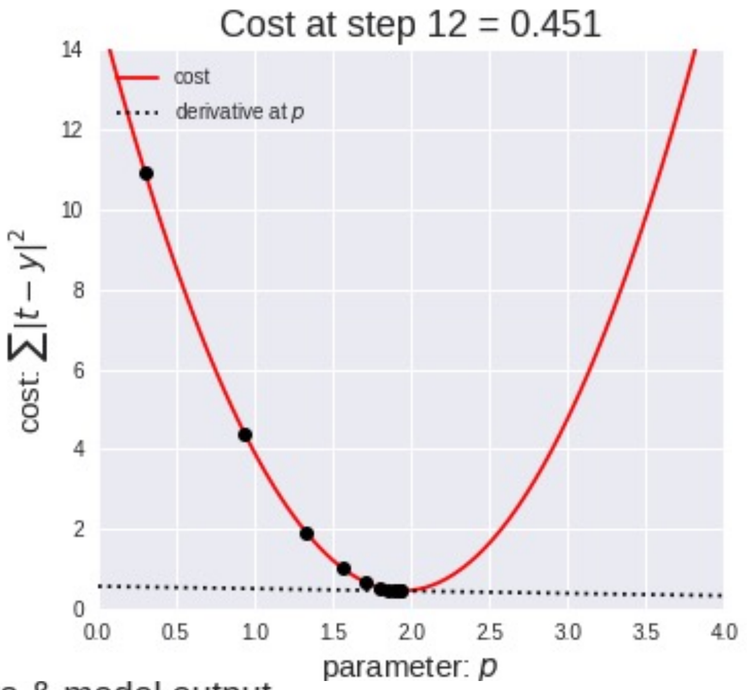
How does training and building the neural network work?

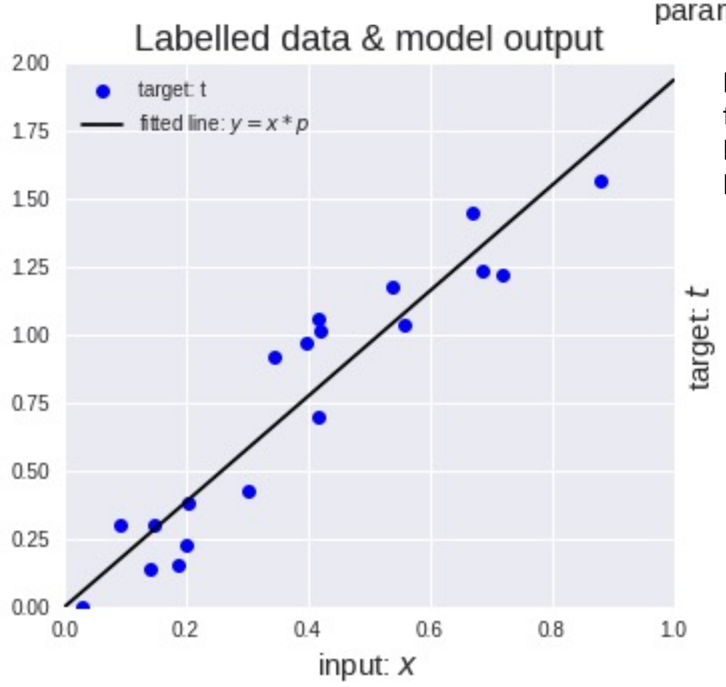
- Divide the labeled dataset that you have into three groups:
    1. Training (~ 60% - 70%)
    2. Testing (~ 20% - 30%)
    3. Validation (remaining data)
        - The validation data are never seen in the training testing period
        - Unfortunately, sometimes the terms of testing and validation are switched
- Pay Attention!!!
    - Many tutorials or examples will randomly split the data into these percentages but in geoscience, we often have spatial and temporal relationships
        - This means that neural networks actually remember images that they've seen before and will associate images with images around the same time and learn from memorization instead of training weights.
    - Typically if there are multiple input types, then the data should be normalized

What are Neural Networks?

Training a neural network model is iterative. Because of this, it is important to understand iterative optimization. Once an input is propagated through the neural network, the output error (or loss) is *backpropagated* to then modify the parameters of the network.

- Gradient Descent: Minimize the *loss function* in the learning portion of training.
  - The *loss function* is the difference between the predicted and actual values.
  - The learning rate, or step size, is seen to be decreasing in this example and the loss function also decreases
  - Neural networks have many parameters, so we are actually looking at many partial derivatives to minimize.





https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

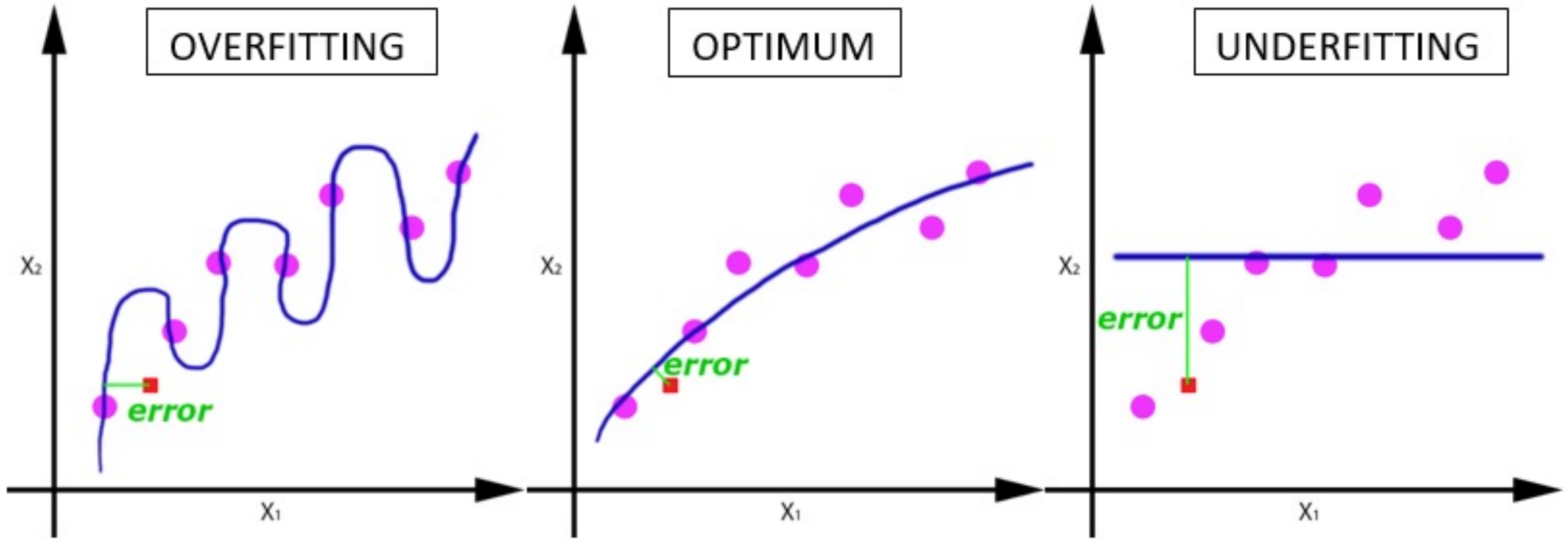
What are Neural Networks?

So why was that important? It’s the whole concept of training a machine learning model.

- Optimizing a neural network is an entire process. There are many aspects of the neural network to consider and parameters to change.
- Changing, or tuning, these is generally referred to tuning *hyperparameters*. These are the dials that you twist and turn to build and tune a model. The variables are set manually before the model is trained.
- We’ll address four very generally:
    1. Epochs
    2. Batch Size
    3. Loss functions
    4. Activation functions

What are Neural Networks?

1) Epochs:
   - *Epoch* is the term used when all the data in a training sample has been cycled through one complete time in the neural network.
   - Basic concept: the more epochs, the more the weights and parameters get changed or the more the model goes from underfitting to overfitting.



https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9

What are Neural Networks?

2) Batch size:

- A *batch size* is the number of training samples fed into the neural network.
- We almost never feed the entire training dataset into the model.
- Select a batch size that will evenly divide the data.
  - If I have 100 samples, I might pick a batch size of 25.
  - You might hear the term iteration, which for our above example would be “4” because it took four iterations of 25 to get through all 100 samples.
- When all training sample batches have been fed into the model, then all training samples have been used to train one epoch.

What are Neural Networks?

3) Loss Functions[4]:

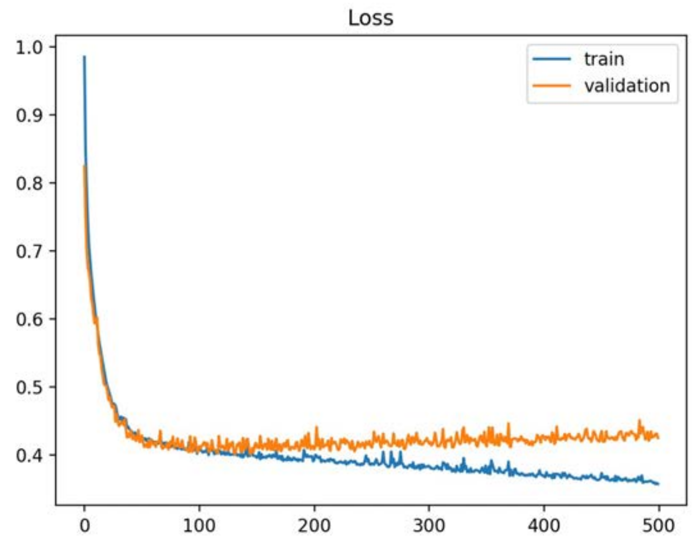
- We have learned that neural networks, and will see subsequently that deep learning, is built off gradient descent learning.
- Gradient descent works by minimizing loss.
- Therefore, we must pick the appropriate loss function to train a model to.
- Common types of loss functions:
  - Mean Squared Error: $\boldsymbol{MSE(x, \hat{x}) = \frac{1}{n}\sum_{k=0}^{n}(x - \hat{x}_k)^2}$ → Sensitive to outliers
  - Binary Cross Entropy: $\boldsymbol{BCE(p, y) = -log(p) \quad if\ y = 1}$
    $\boldsymbol{BCE(p, y) = -log(p) \quad if\ y = 1}$
    → Weighs loss to different instances and penalizes specifically if model guessed wrong
  - Tversky/Dice: $\mathbf{T}\boldsymbol{C} = \frac{\boldsymbol{|X \cap Y|}}{\boldsymbol{|X \cap Y| + \alpha|X - Y| + \beta|Y - X|}}$ → Weighs loss to specific instances based on specific $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ weights

What are Neural Networks?

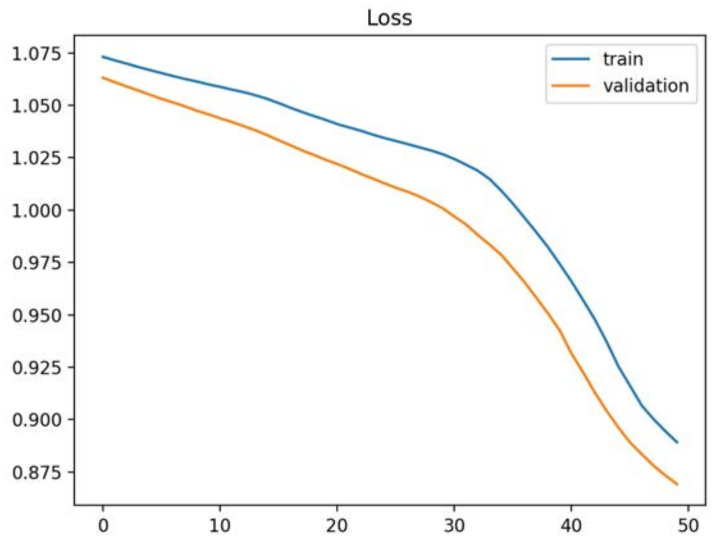
3) Loss function[4]:

- Looking at the plot of loss in relation to the number of epochs for both the training and testing dataset is also incredibly useful

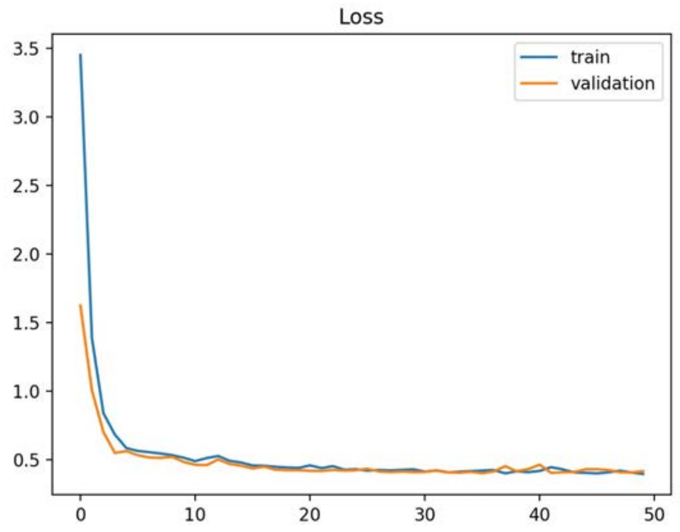
https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/



What do we think?
This shows that the model is overfit!



What do we think?
This shows that the model is underfit!

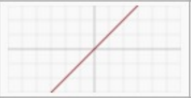
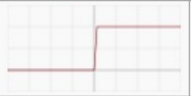
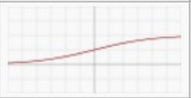
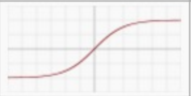
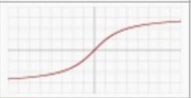
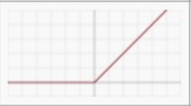
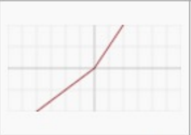
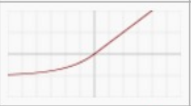
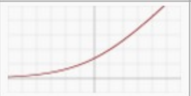


What do we think?
This shows that the model is just right!

What are Neural Networks?

4) Activation Functions[5]:

- Wait, we can change the loss function but now you're saying there are different activation functions, too?
- These are basically transfer functions that keep the neural networks moving forward by collecting everything from a neuron and then reducing it to a single value for the next step in the model network.
- It can be liner or non-linear.
- It will (maybe) make more sense in the next section.

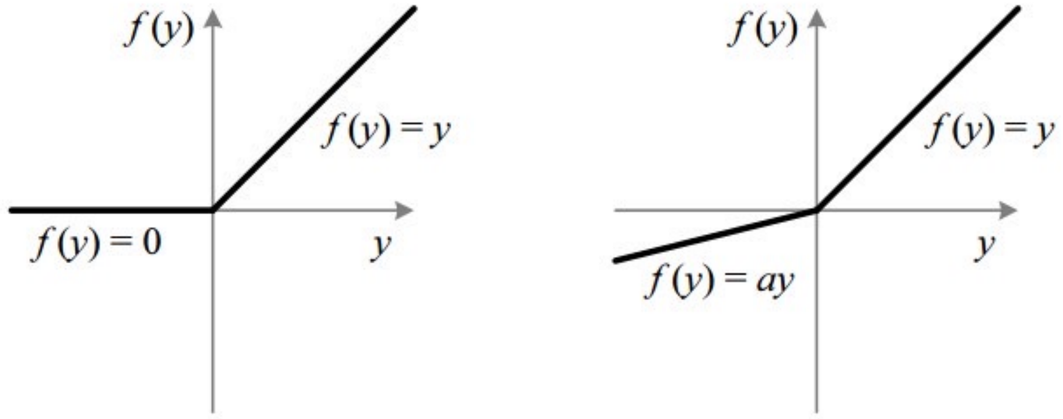
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

What are Neural Networks?

4) Activation Functions:

- That table was actually missing “Leaky ReLu”
- This will come back in the next section because it’s one of the more commonly used function since it allows values to range from $\pm\infty$



https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

What are Neural Networks?

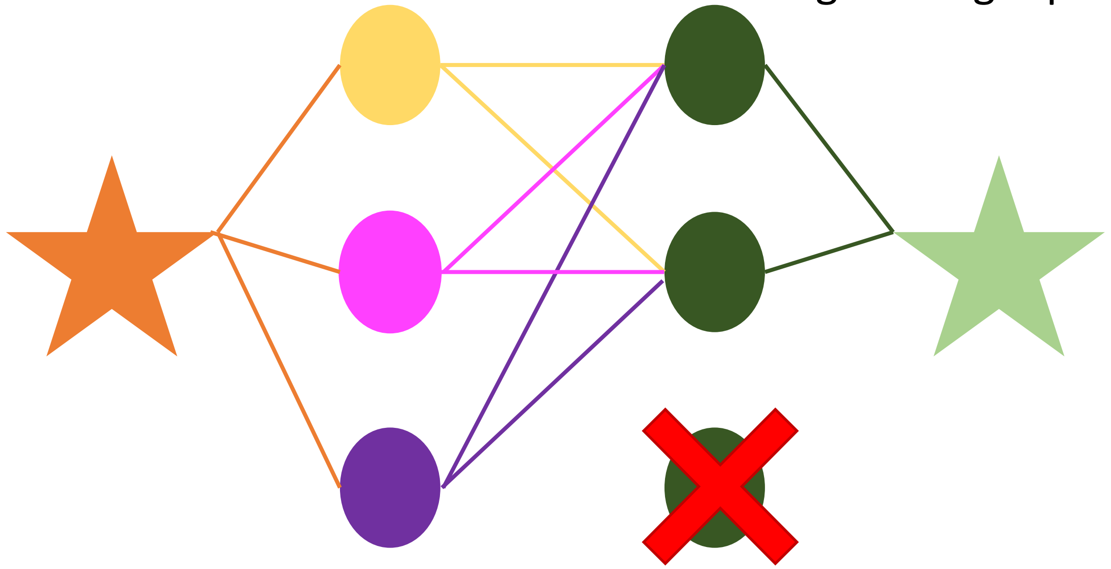
How can one further avoid over-fitting data?

- Check for imbalanced dataset problems
  - These can lead to fitting the more frequent event
  - Question: Can you think of some common imbalanced dataset challenges in your field of research?
- Add *noise* into the inputs
  - Noise is added to the samples so that each training sample is less likely to be memorized as a "hey I've seen this input before!"
    - This type of issue can be really common for geosciences and often noise will help with the model memorizing inputs instead of learning the features.

What are Neural Networks?

How can one further avoid over-fitting data?

- Add dropout into the network
  - *Dropout* will drop a randomly selected percentage of neurons at a given layer in your model.
    - Dropout percentages too low will have minimal or no effect on training
    - Dropout percentages too high will lead to under learning in the model training
    - Dropouts should be used with a model that has a large enough quantity of neurons to drop



So what is deep learning?

Woof... Let’s take a break to digest all the specific terms in basic neural networks by pulling back to more general concepts and then diving into the structure of a deep learning model.

What is deep learning?

- It is literally a neural network that has multiple "hidden layers."
- These hidden layers begin to build up and look at deeper and deeper connections that might solve the initial, simpler problem.
  - Deeper and deeper connections
  - What makes a number a number?
    - Curves, contrast, size, edges, etc.
  - What makes a face a face?
    - Does it have eyes? Does it have a mouth? Does it have teeth?
  - What is a horcrux?
    - Why could a diary control a girl? Who is Tom Riddle? Is the connection of speaking snake important? What is his end game? Why is he obsessed with Harry? Why did he need the journal to live?

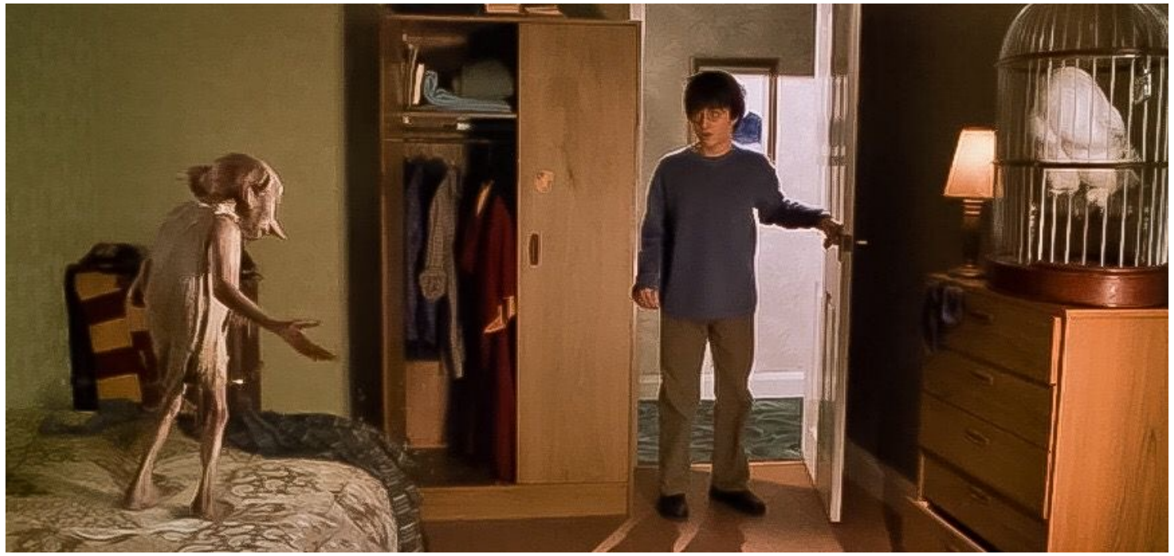
Exercise: Think of a classification problem and set it up where you'd start with simpler identifying features and then get more complicated.

What is deep learning?

Last bit of housecleaning: While deep learning has many applications, some of which don't involve images, when deep learning is used with images it can be confusing.

Image classification

- Produces a list of labels with what the image may contain ([0,1] where 1 = contains, 0 = not in image)

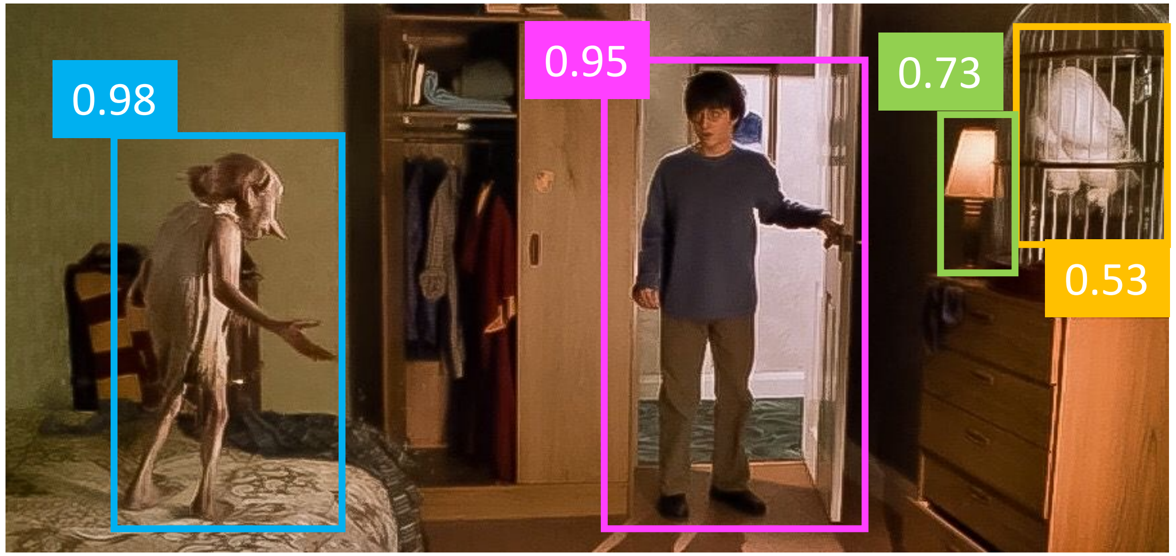


1. Human: 0.97
2. House elf: 0.82
3. Lamp: 0.66
4. Owl: 0.51

What is deep learning?

Image detection

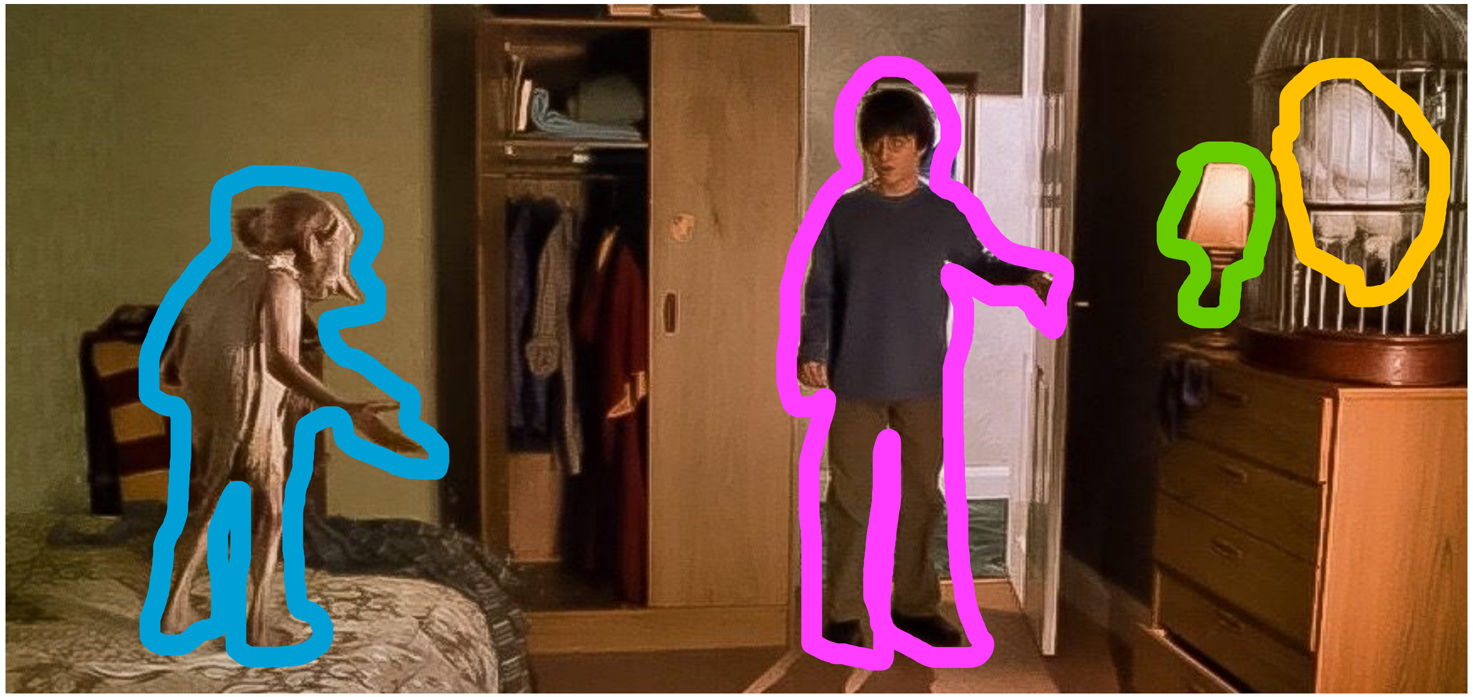
- Considered a “computer vision” problem
- Looks within an object to classify specific objects
- Think “bounding boxes” around objects, each with their own assigned value [0,1]
  - These values are often related to “confidence” of an object but this is confidence in the model’s output, not necessarily direct statistical significance. For instance a model can give the value of 1 but it doesn’t mean the prediction is 100% correct.



What is deep learning?

Image segmentation

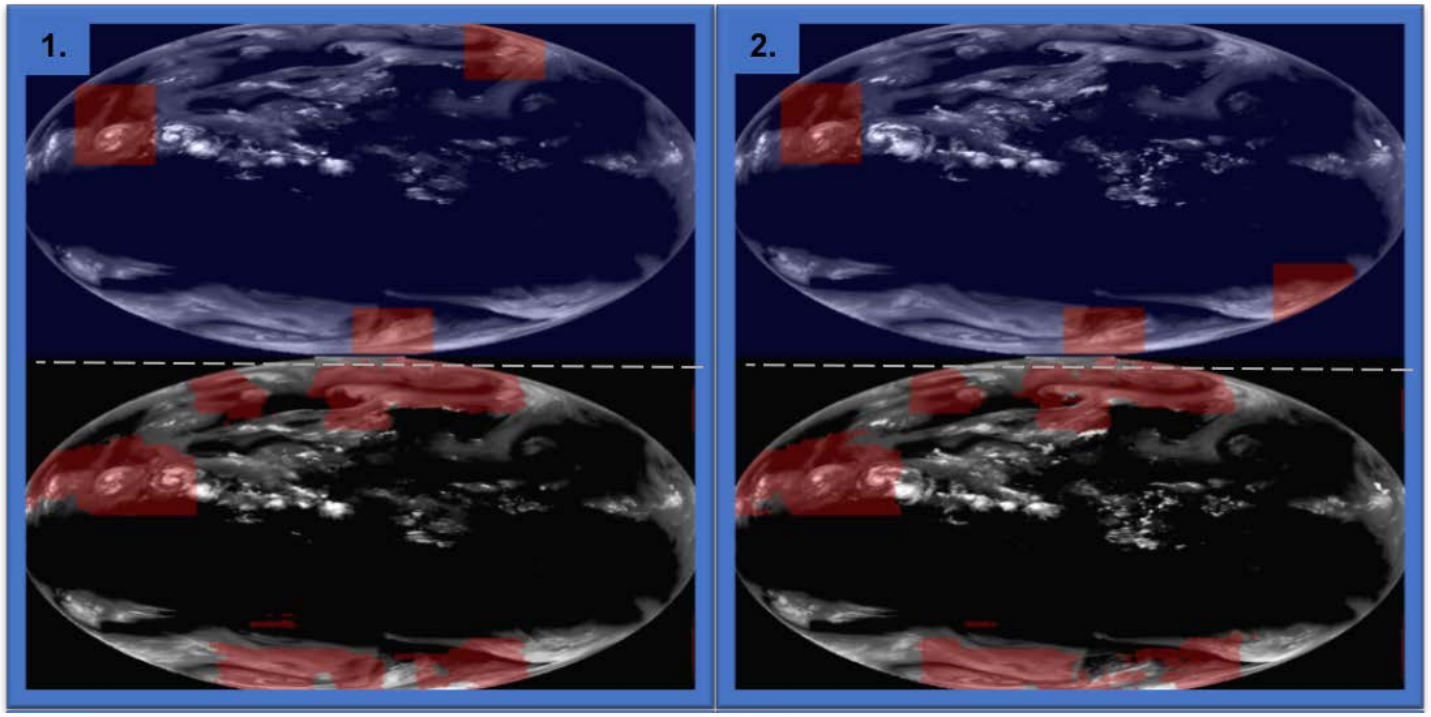
- This is related to detection, however every pixel within an image is given a label
- This allows for the entire object boundary to be labeled, as each pixel has a label ranging [0,1].



What is deep learning?

Image segmentation

- For my own cyclone work, an entire image is fed into the network and then it looks at the input image pixel by pixel to assign a confidence value per each pixel that the object is included in that pixel



I chose to create a square label to put over the cyclone, it was the easiest way to create labels, however it's each pixel that has the value of "1."

The U-Net model's prediction of which pixels contain a cyclone. I applied a threshold of "pixel must be equal to or greater than 0.7"

What are Neural Networks?

More ingredients and buzz words of neural networks and deep learning you should look out for in upcoming slides. When these come up, write them down:

- *Encoder*
- *Decoder*
- *Fully Connected Layer*
- *Dense Layer*
- *Convolutional layer*
- *Stride*
- *Rectifier*
- *Kernel*
- *Feature map*
- *Pooling*
- *Up-convolutions*
- *Decoder*
- *Skip connection*
- *Zero padding*

Christina’s Example of Deep Learning

The Problem:

- From Global Forecasting System (GFS) total precipitable water output or Geostationary Environmental Satellite System (GOES) water vapor imagery, train a deep learning model to locate tropical or both tropical and extratropical cyclones within the image.
- This is a classic “image segmentation” problem where an entire image is input into a deep learning model and the model return a pixel-by-pixel segmented image. In this case, there is only one classification of “0: no cyclone” and “1:cyclone”
- This entire project probably took around 1.5-2 years. Data collection, labels, training, retraining, etc.

Tropical and Extratropical Cyclone Detection Using Deep Learning (https://journals.ametsoc.org/view/journals/apme/59/12/jamc-d-20-0117.1.xml)

Christina's Example of Deep Learning

Datasets:

- Global Forecasting System (GFS) total precipitable water output
- Geostationary Environmental Satellite System (GOES) water vapor imagery

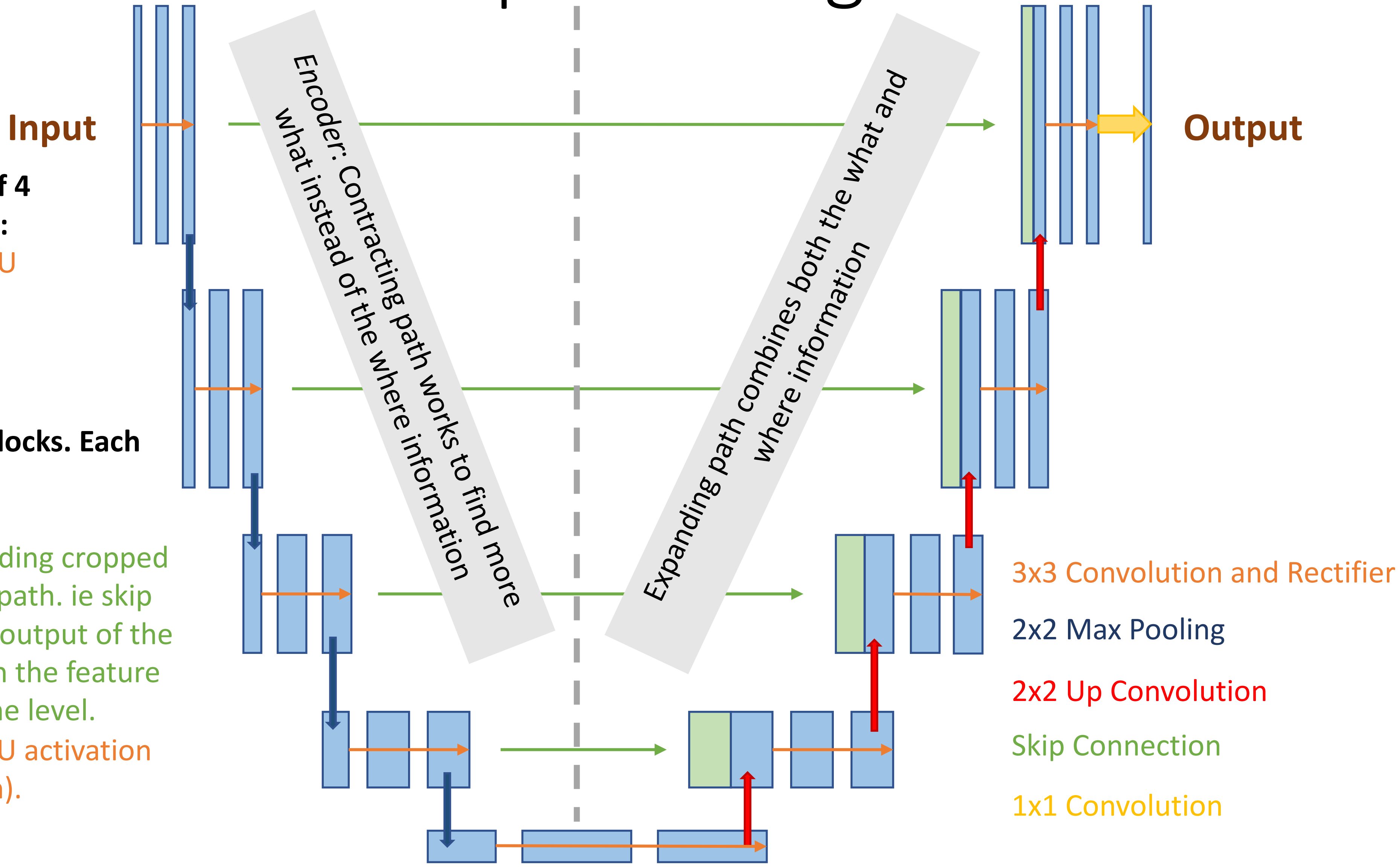
Dataset Labels:

- IBTrACS[1]
    - Tropical cyclone International Best Track Archive for Climate Stewardship
    - Biases?
- Heuristic-created cyclone and tropical cyclone labels[2]
    - Heuristic model developed for labeling cyclones for this specific purpose
    - Biases?

[1]Knapp, K. R., Kruk, M. C., Levinson, D. H., Diamond, H. J., and Neumann, C. J., The International Best Track Archive for Climate Stewardship (IBTrACS): Unifying tropical cyclone best track data. Bulletin of the American Meteorological Society (2010), 91, 363-376. non-government domain doi:10.1175/2009BAMS2755.1

[2]Bonfanti, Christina, et al. "Machine Learning: Defining Worldwide Cyclone Labels for Training." 2018 21st International Conference on Information Fusion (FUSION), 2018, doi:10.23919/icif.2018.8455276.

The Infamous U-Net Deep Learning Model


Input
Output
Encoder: Contracting path works to find more what instead of the where information
Expanding path combines both the what and where information
3x3 Convolution and Rectifier
2x2 Max Pooling
2x2 Up Convolution
Skip Connection
1x1 Convolution

The encoding path is composed of 4 blocks. Each block consist of [7] **:**

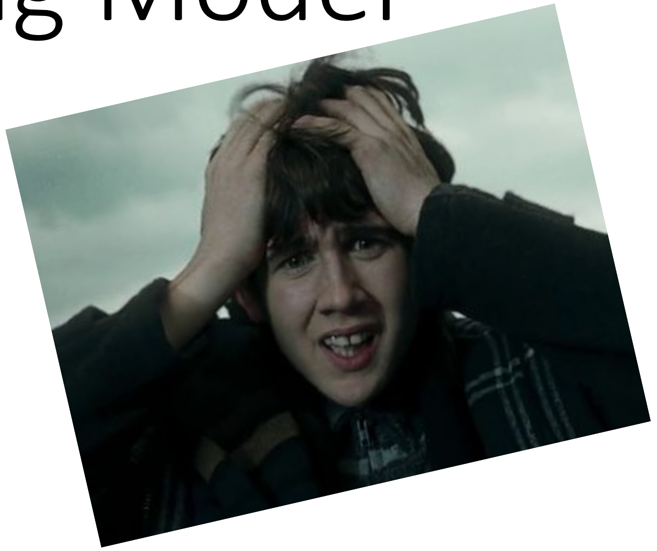
- Two 3 x 3 convolution layers + ReLU activation function (with batch normalization)
- One 2 x 2 max pooling layer.

The expansion path consists of 4 blocks. Each block consists of [7] **:**

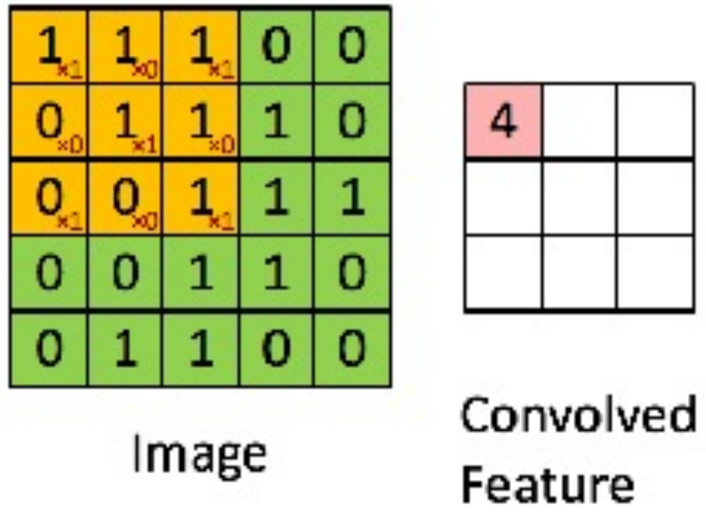
- Deconvolution layer with stride 2.
- Concatenation with the corresponding cropped feature map from the contracting path. ie skip connections by concatenating the output of the transposed convolution layers with the feature maps from the Encoder at the same level.
- Two 3 x 3 convolution layers + ReLU activation function (with batch normalization).

The Infamous U-Net Deep Learning Model

Convolutional layer: Dot product of entries and a filter to produce an activation map that shows the impact of one input on the other's shape



A convolution is a process where some filter performs convolutions on the input matrix. This filter is typically $2x2$ or $3x3$ (in our case). It slides along the input up and down with some *stride*, or how many pixels per step, as well as side to side.



The *rectifier* is what you already know as the activation function. The convolution and rectifier are done same-process. First the rectifier is applied to each pixel and then the dot product is calculated. If it helps, the rectifier is applied as the convolution is happening to make changes (or maybe contrasts) more clear.

Pixel value and weight come in, passes through activation function, then moves onto the convolution.

https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

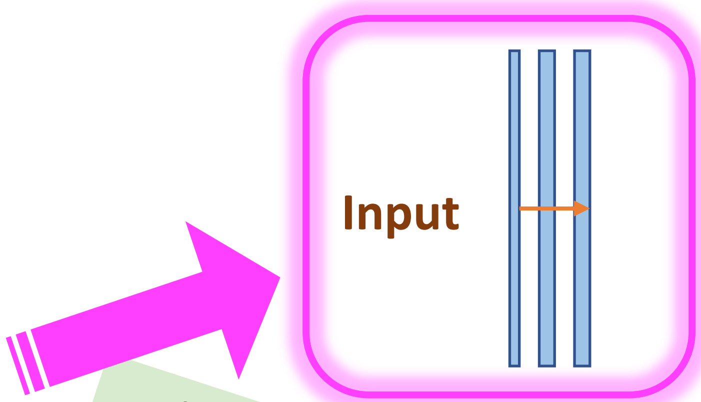
The Infamous U-Net Deep Learning Model

Example working with an input image sized: $205x205x1$

Here we have $205x205x1$ input and if our model has 64 *kernels* (or, 64 of the $3x3$ filters) then once we are done with the convolution, I have a *feature map* of $205x205x64$ (remember, a $3x3$ means I hit the side and lose 2). The feature map is the result when features from one space more to another.

A kernel performs the 2D convolution on the entire 3D at one time. So by doing this 64 times, we keep the third dimension at 64.

Now there is one more convolution at this layer. It's not necessary, and I could do it even more times, but twice is typically how it is done in most U-Nets. At the end of this layer, we have a feature map size $205x205x64$ moving through the network.



Side note: you're likely to hear about *fully-connected* layers as well as convolutional. These are different in how they connect to other neurons. Fully connected are, in fact, connected to every neuron (every neuron gets the output from this neuron) whereas fully convolutional are connected to local neurons and more efficient. Conversely, the *dense* layer is fully connected from every network (every neuron gives its output to this layer)

The Infamous U-Net Deep Learning Model

Don’t forget the whole purpose of deep learning:

We are training a model to extract features within the input image that are important in identifying our desired object(s)/region(s).

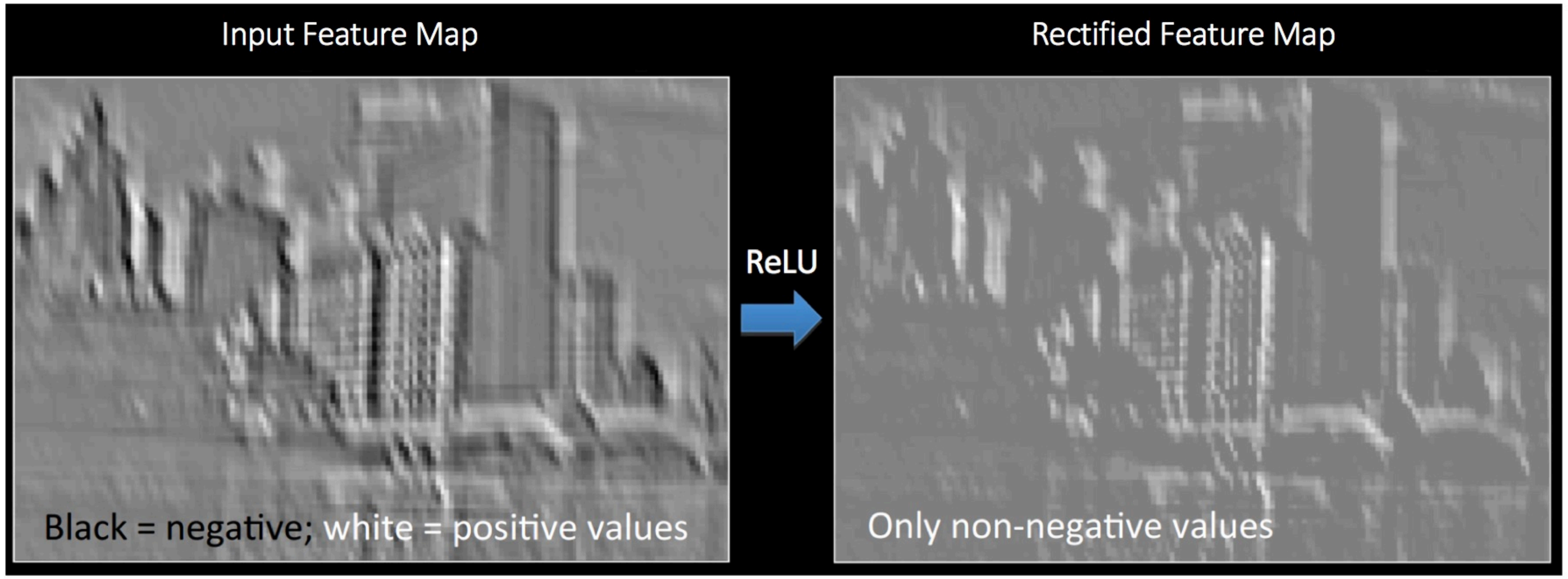
https://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html/2

In photography terms, we take a high resolution image and make it fuzzy on purpose to identify the important contrast spots or lines or boundaries.

The Infamous U-Net Deep Learning Model

Don't forget the whole purpose of deep learning:

We are training a model to extract features within the input image that are important in identifying our desired object(s)/region(s).



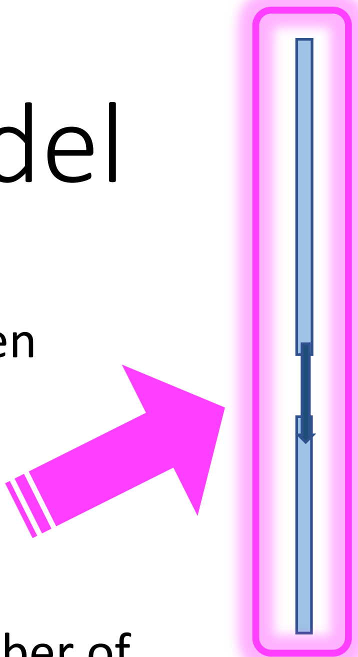
https://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html/2

In photography terms, we take a high resolution image and make it fuzzy on purpose to identify the important contrast spots or lines or boundaries.

The Infamous U-Net Deep Learning Model

Pooling layer: Gathers outputs of the previous neuron clusters at layer L-1 and then combines it into a single input for the next layer L.

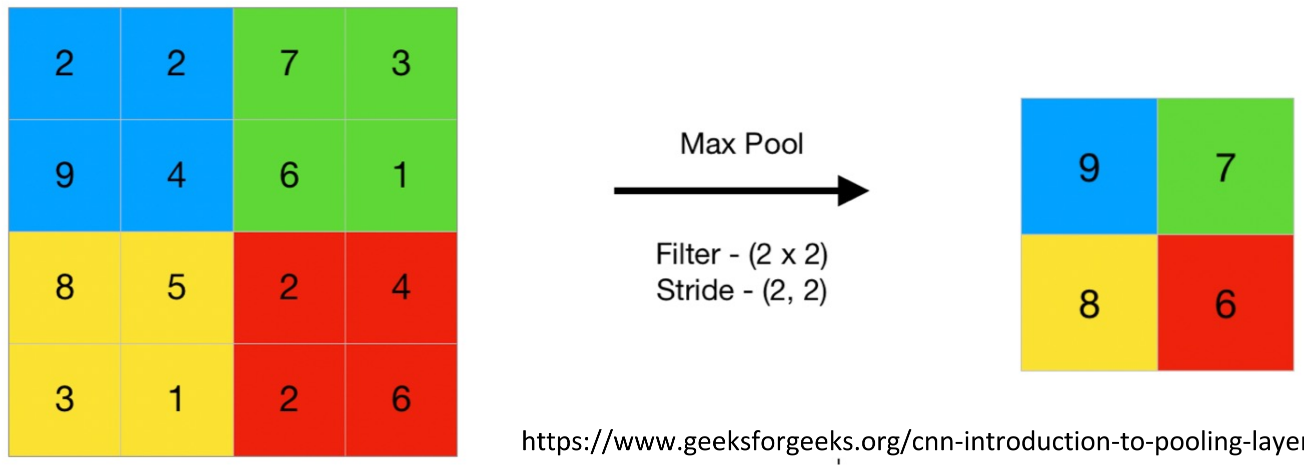
Max Pooling: Takes the max output value of each of the previous cluster at L-1



Pooling occurs to reduce the spatial dimensions. By reducing the dimensions, the number of parameters that the model has to is reduced and therefore so is the computation.

In our U-Net, after completely the second convolution where its dimensions are $205x205x64$, the pooling layer decreases the dimensions to $102x102x64$

Did you notice something funky in the math? If you have an odd numbered integer going into a pool layer, it rounds the division down.



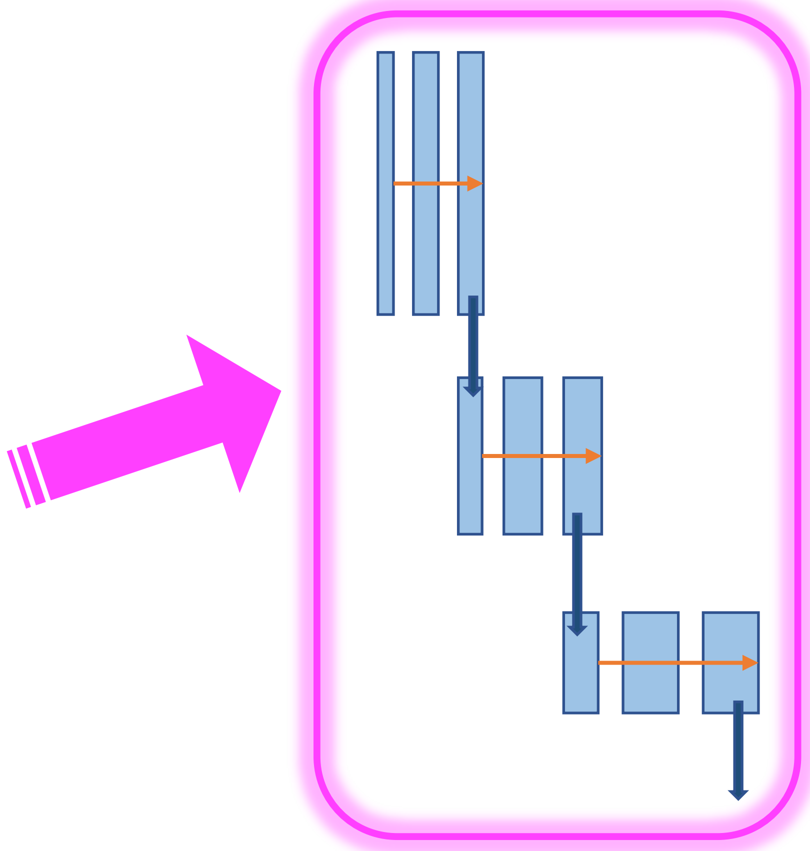
https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

The Infamous U-Net Deep Learning Model

Now, down to the next layer. Again, there are two convolutions and then a pooling layer.

These downward layers are what captures the contextual information important to deciding if the object in question will exist in a pixel or not.

In this example, we go through a total of four layers. The first we covered a few slides ago, and then three more.



The Infamous U-Net Deep Learning Model

For our example, recall the input into layer two was size:
$205x205x64$

The size of the input feature map at the end of Layer 2: $102x102x128$

This is because $205/2 \ = 102.5$ then round down to 102

This is also because $64 * 2 \ = \ 128$

The size of the input feature map at the end of Layer 3: $51x51x256$

This is because $102/2 \ = 51$

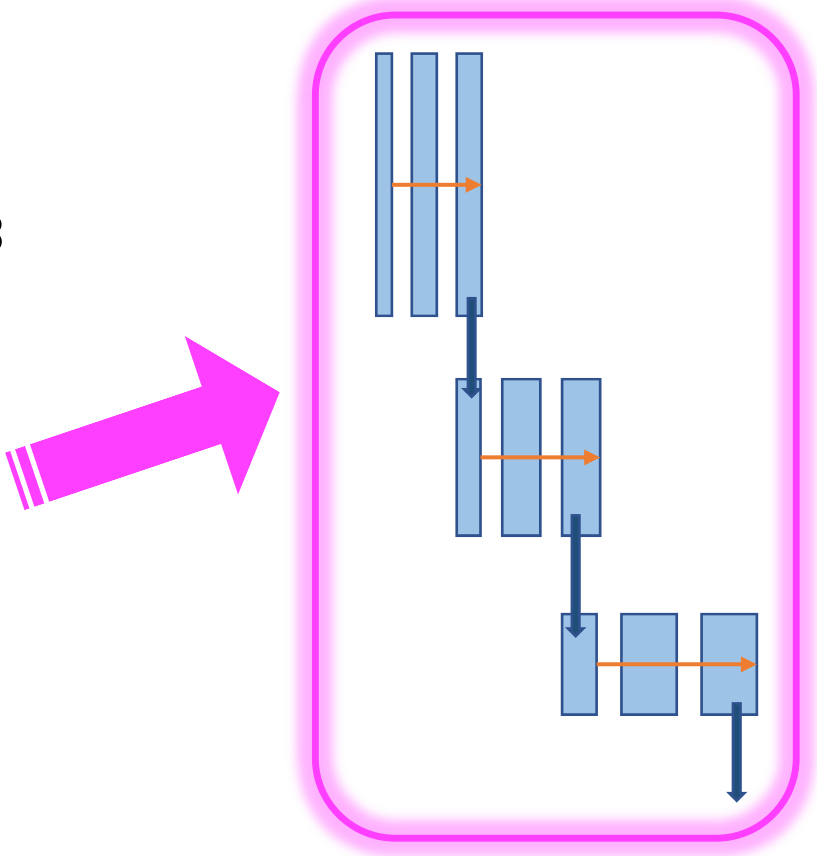
This is also because $128 * 2 \ = \ 256$

The size of the input feature map at the end of Layer 4: ?

Did you say $25x25x512$?

This is because $51/2 = 25.5$ then round down to 25

This is also because $256 * 2 \ = \ 512$



The Infamous U-Net Deep Learning Model

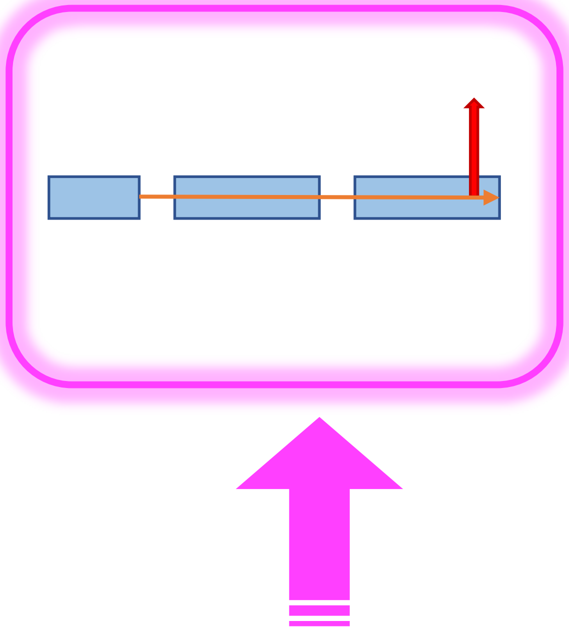
Let’s call this the bottom of the U-net structure. After the image in layer 4 goes through the max pooling, we have two more convolutions at bottom layer 5.

The size of our example after these two convolutions will become:

$$12x12x1024$$

But then what special is happening?

Up-convolutions or *up-sampling* or a *transpose convolution*: Resizing (increasing) the image by copying pixel values. This is pretty much the opposite of pooling.

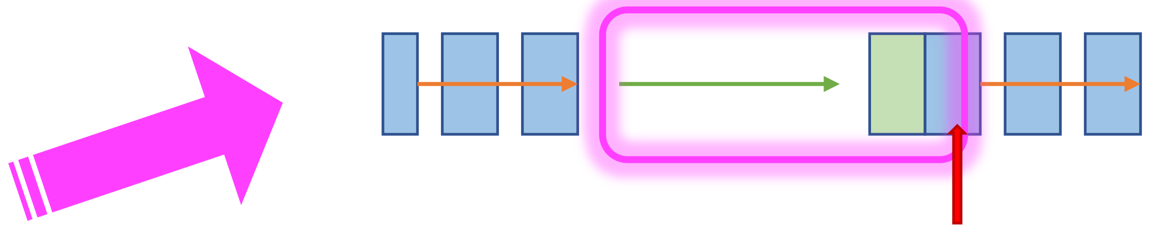


The Infamous U-Net Deep Learning Model

We up-sample and merge information in the skip connection stage in order to recover spatial information from the input. As we move back upwards, this is referred to as the *decoder* portion of the network.

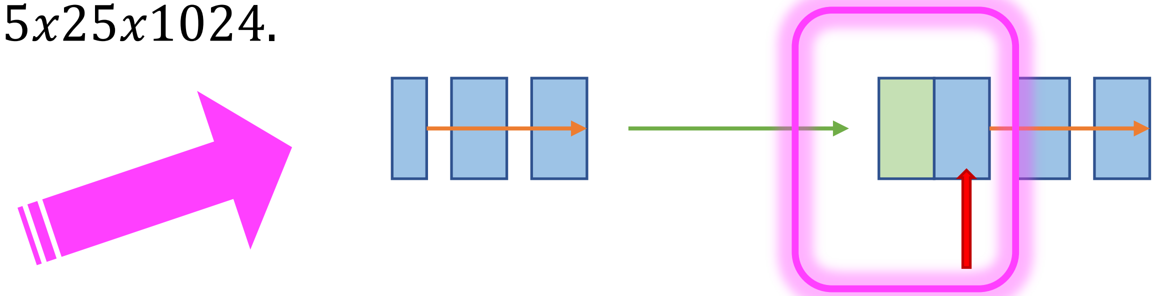
Working back up the network with up-convolutions also uses the most unique feature about the U-Net: skip connections.

Skip-Connections: Unique to the U-Net, these connections further improve the information of precise locations by concatenating the output of the up sampling convolution layers with the feature map (convolution) from the corresponding downward layer.



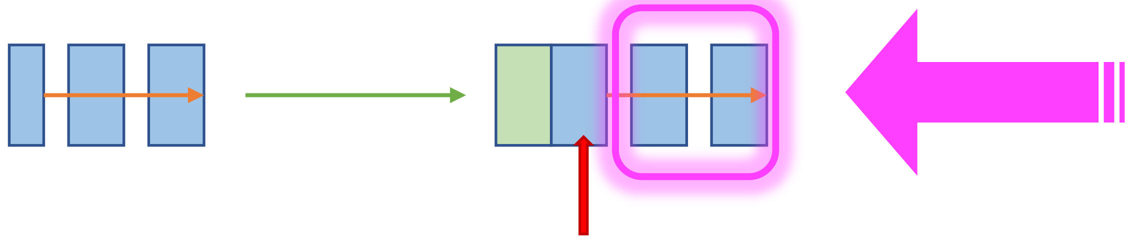
The Infamous U-Net Deep Learning Model

Once we run up-sampling on our bottom layer, our example's dimension increases from $12x12x1024$ to $25x25x512$. This is immediately concatenated in the skip-connection, which increases the filter dimension to $25x25x1024$.



Recall: Each filter actually convolves over all existing filters. For instance, it is convolving on the first 2D but also to a depth of however many filters have been stacked to the image. Therefore, during these new convolutions, we can actually reduce the third dimension by only performing half the filters used on the encoder convolutional stages.

Two more convolutions are preformed to the image to reduce the third dimension, which reduces the number from 1024 back to 512. The final feature map size in this layer 4' is sized: $25x25x512$.



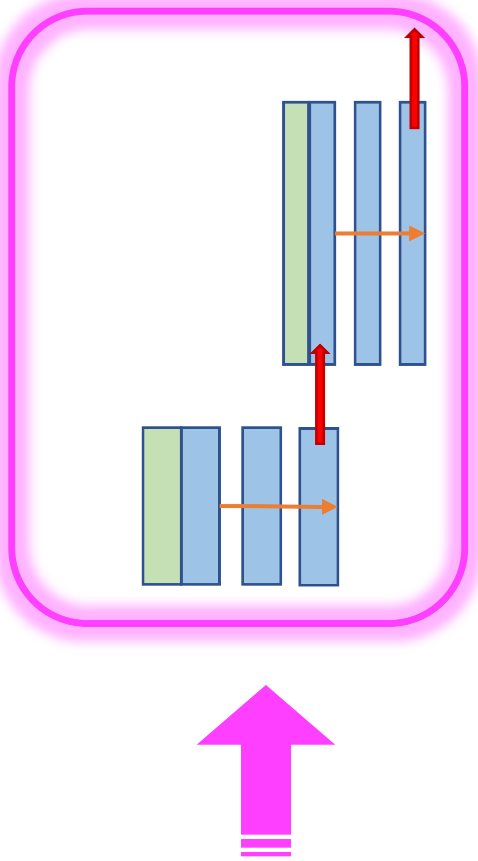
The Infamous U-Net Deep Learning Model

There are two more additional layers, matching the same number as the encoder side of the model, of up- sampling and convolutions.

The size of the input feature map at the end of Layer 3’: $51x51x256$

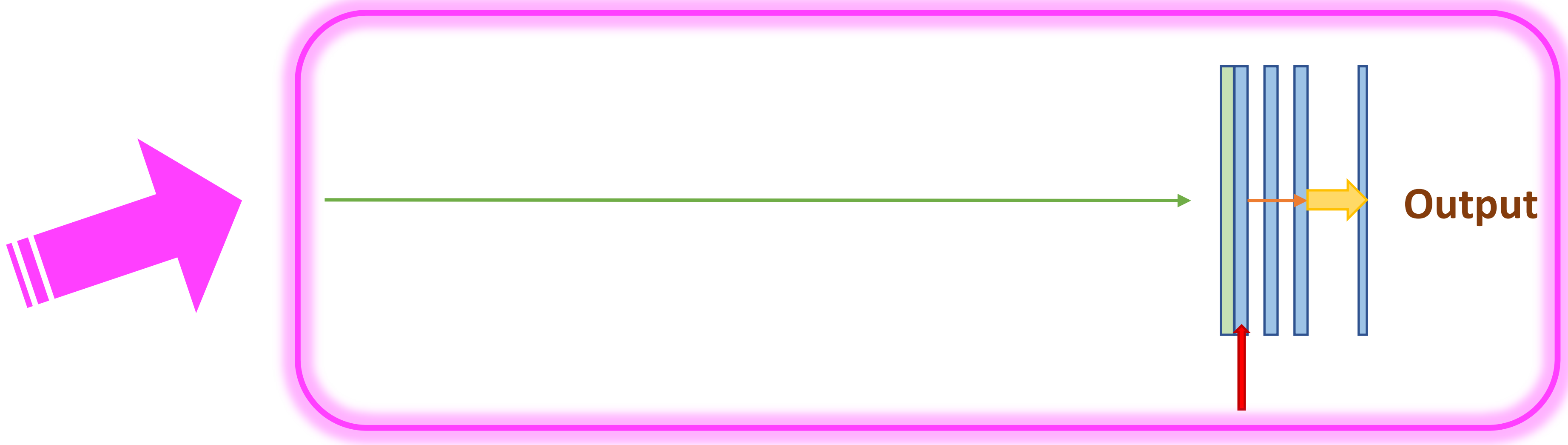
The size of the input feature map at the end of Layer 2’: $102x102x128$

Note: when up sampling back to an odd number, we can use *zero padding*, which is adding one row and one column of zeros



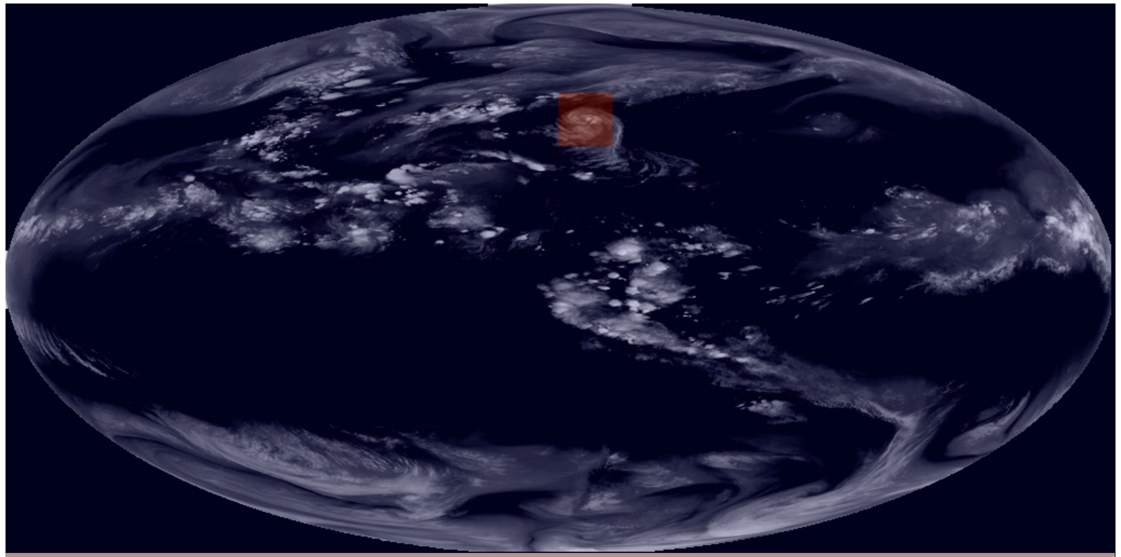
The Infamous U-Net Deep Learning Model

The last layer, layer 1', will produce the segmented image map back to same size as the input image. This involves one additional convolution with filter size of 1 so that the model produces an output the same size as the input image: $205x205x1$

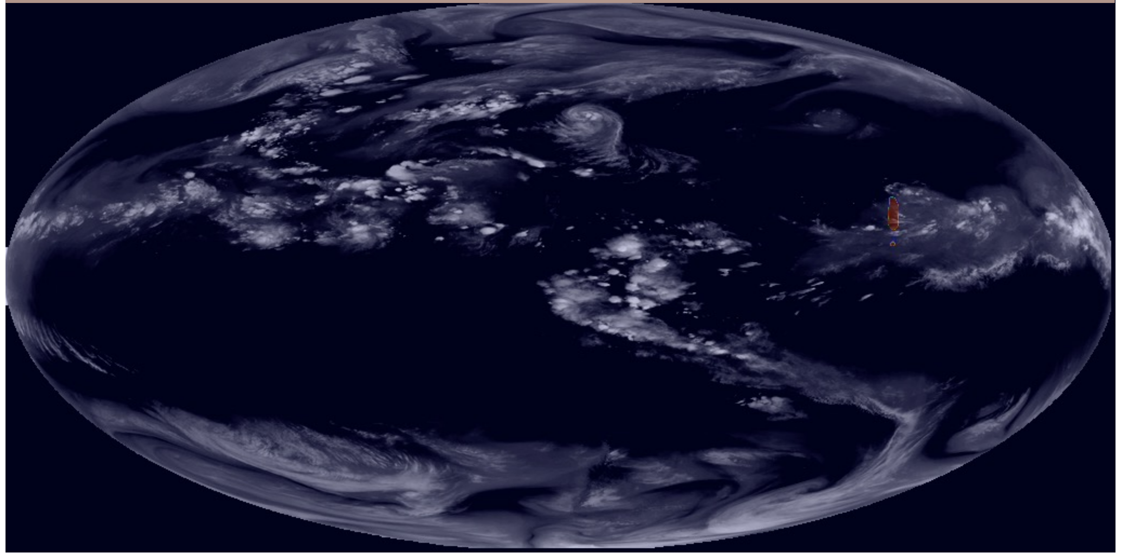


Christina’s Example of Deep Learning

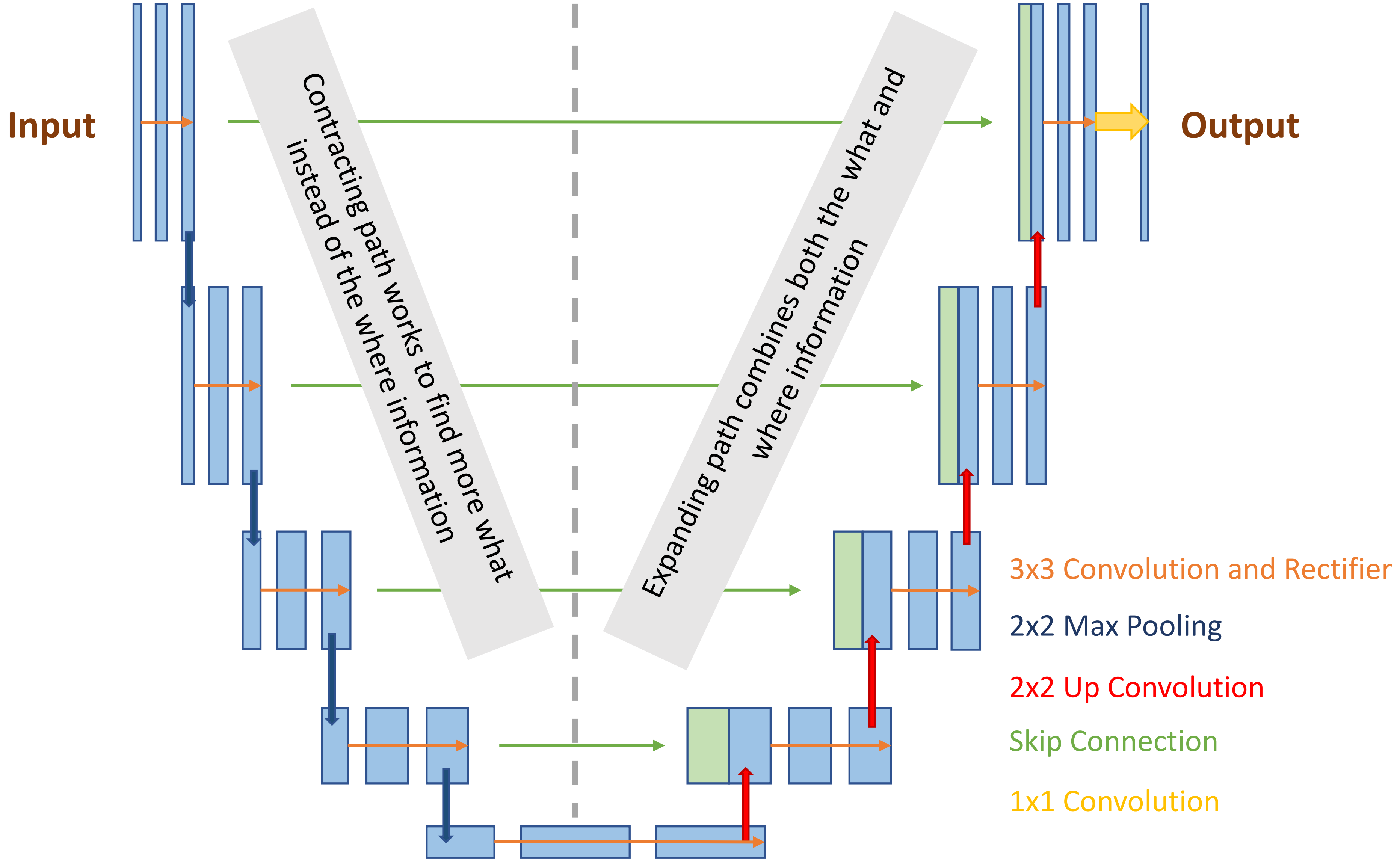
Truth Label:



UNET Label:



The Infamous U-Net Deep Learning Model


Input
Contracting path works to find more what instead of the where information
Expanding path combines both the what and where information
Output
3x3 Convolution and Rectifier
2x2 Max Pooling
2x2 Up Convolution
Skip Connection
1x1 Convolution

Christina’s Example of Deep Learning

UNET Model Architectures													
Model Labels	Model Input	Input Image Size	Training Size	Validation Size	Batch Size	ROI Label Size	Dropout or Noise	Loss Function	UNET Model Depth	Epochs to Convergence	Training Time (seconds)	Inference Time per Single Run (seconds)	Inference Time per Month of Runs (seconds)
IBTrACS	GFS	720x361	8622	3020	256	25	noise 0.2	Focal	6	37	2130	0.03	8.16
Heuristic	GFS	720x361	15574	2902	1520	30	dropout 0.1	Dice	5	200	2200	0.03	6.48
IBTrACS	GOES	1024X560	5638	2214	128	25	dropout 0.2	Binary Cross Entropy Dice	5	70	3540	0.15	36
Heuristic	GOES	1024X560	25288	2735	720	60	dropout 0.1	Tversky	4	150	4650	0.06	14.4

Christina’s Example of Deep Learning

UNET Model Results								
Model Labels	Model Input	Accuracy	Loss Score	Dice Coefficient	Tversky Coefficient	Optimizer	Dropout or Noise	Batch Normalization
IBTrACS	GFS	0.991	0.237	0.763	0.750	rms 0.00008	noise 0.2	yes
Heuristic	GFS	0.807	0.351	0.58	0.649	rms 0.00001	dropout 0.1	yes
IBTrACS	GOES	0.996	0.311	0.689	0.680	rms 0.0001	noise 0.1	yes
Heuristic	GOES	0.901	0.442	0.511	0.558	rms 0.00001	dropout 0.1	yes

We looked at the *Tversky coefficient* to measure how well our model performed against our truth because this compares “likeness” as opposed to accuracy’s binary comparison. We set $\alpha = 0.3$ and $\beta = 0.7$:

$$\frac{|X \cap Y|}{|X \cap Y| + \alpha|X - Y| + \beta|Y - X|}$$

I highly recommend referencing [6] to gain insight on the trade-offs of certain confidence and performance metrics.

Christina’s Example of Deep Learning

Final thoughts:

- How do we actually measure success?
  - Is it in the numerical results?
  - Is it in the pictures?
  - ➢ I argue it depends on the application of the problem
    - Comes back to what problem is this machine learning method trying to solve
  - ➢ It must be measured in relation to the labels that trained the model
    - A model will only ever be as good as the labels provided
- What can be learned from the UNET outputs
  - Understand which features were most important to determining the ROI
    - Is there things meteorologists don’t yet understand in certain model or satellite channels that indicate an ROI?

When should we use deep learning?

There are many opportunities to use ML, and it’s been increasingly popular in the geosciences given our access to tremendous amounts of data, however deep learning should be used with caution.

When should we use deep learning?

The most important rule about using machine learning:

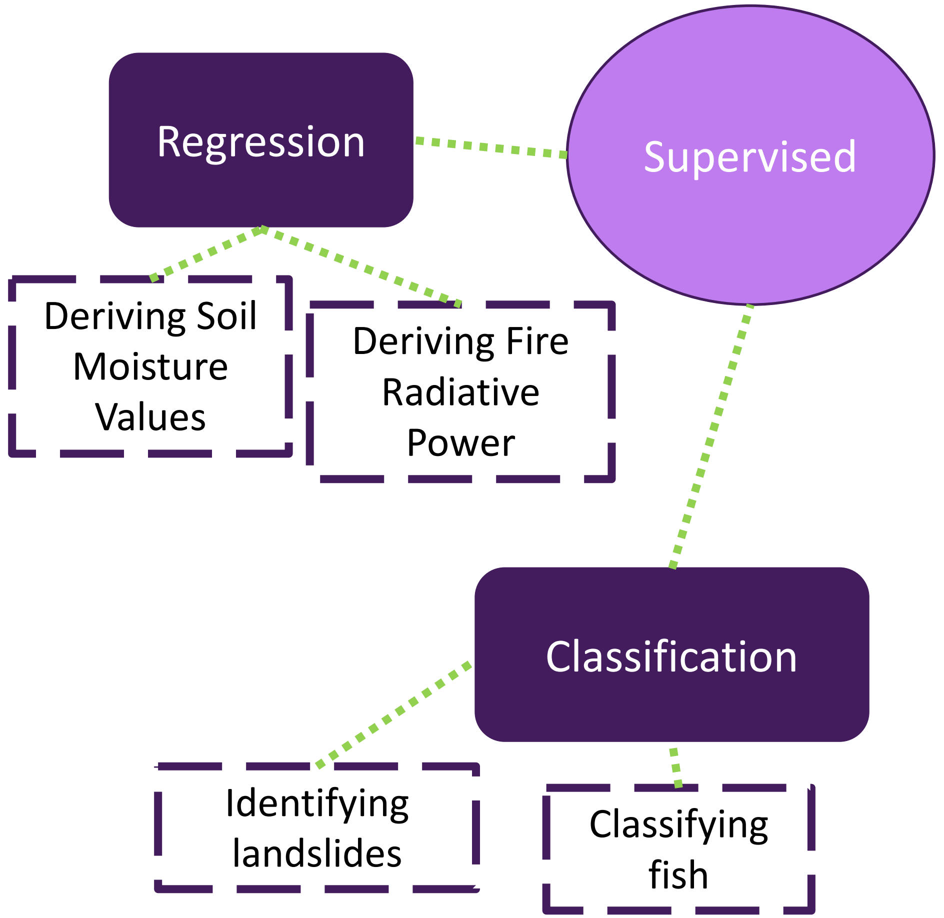
If it can be solved/done with a simpler model, use the simpler model!

Don’t rush in head first. Instead think it through. Try:

1) A regression model
2) A single layer neural network
3) Shallow lay deep learning --> while possible, and sometimes necessary, it is typically advised to keep the number of hyper parameters similar to the number of input samples

When should we use deep learning?

Recall all that neural networks and deep learning might be suited to do!



Deep learning can be used in many different types of applications:

- Emulating and accelerating radiative transfer models
  - This one is in house with GSL's own Ryan Lagerquist
  - https://journals.ametsoc.org/view/journals/atot/38/10/JTECH-D-21-0007.1.xml
- Satellite image segmentation
- Predicting climate (ENSO, PDO, etc.)
- Parameter estimation
- Quality control and model verification
- Atmospheric chemistry
- Hurricane Intensification
  - Work actively done by CIRA neighbors

When should we not use deep learning?

Be a Hermione and think it through.

When should we use not use deep learning?

- Again, the first rule is to not dive into deep learning.
- Think about the amount and diversity of data that you have to train a model.
- Think about the type of problem you're trying to solve.
    - Are there incredibly complicated, non-linear relationships or is my problem more general?
- Is deep learning breaking the rules of physics?
- Biases in datasets or models.

Some of these issues can be improved by using up-and-coming methods such as explainable AI and physics-based AI methods.

THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Xkcd.com

When should we use not use deep learning?

- Machine learning will not ever completely replace scientists
  - Scientists are needed throughout the entire model development.
- Never trust a paper that doesn’t share multiple metrics of measuring success.
  - For example, if there’s a model that has a 98% accuracy at detecting hurricanes, think about what that means. How often are non-hurricane moments used in the training? How about hurricane? Then is the model’s accuracy skewed to non-hurricane detections?

Brain Exercise!

Brain Exercise!

Take 5 minutes to practice setting up your own deep learning problem.

Either on your own or in small groups, join up and take five minutes to think of a well-suited deep learning science problem. Plan out how you might set up using deep learning for this problem. Things to consider:

1. Data inputs
   - Where does it come from?
2. Data Labels
   - Where do these come from?
3. How much data you have
   - Years? Is it minutely data? How expensive is it to collect and label the data?

Additional Resources

Highly recommended tutorials and blog posts

Additional Resources

[1] Neural Networks and Deep Learning
- Online (free) book: http://neuralnetworksanddeeplearning.com/chap1.html

[2] Scikit-learn’s homepage
- Python’s list of very helpful ML-specific packages: https://scikit-learn.org/stable/

[3] MIT’s additional recommended course/tutorial
- http://introtodeeplearning.com/

[4] Guide to selecting loss functions based on ML models
- https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

[5] Guide to the differences in activation functions
- https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[6] Great guide to confidence skills and analyzing performance of deep learning segmentation models
- https://towardsdatascience.com/how-to-use-confidence-scores-in-machine-learning-models-abe9773306fa

[7] A guide I like to the U-Net deep learning model
- https://medium.datadriveninvestor.com/an-overview-on-u-net-architecture-d6caabf7caa4