Deep Learning Presentation

// Intro //

* Everyone here is already familiar with how popularized artificial intelligence has become over the last year and how AI is being integrated into society.
* It’s evident that AIs are becoming increasingly more intelligent and that their ability to solve complex problems improves each and every day.
* I’m going to be talking about how this is made possible by discussing a learning model for AI called deep learning.
* Here’s an outline for my presentation
* First, I’lll discuss what deep learning is and how it’s used in different applications. Next, I’ll talk about how deep learning works.

Then, I’ll talk about the future of learning models and where artificial intelligence is headed.

// Background / History //

* Deep learning is a form of machine learning which aims to solve problems the way humans do
* Artificial intelligence utilizes deep learning to take very complex data and recognize patterns within similar sets of data
* Tim Dettmers, one of the developers working for Nvidia, wrote an article on the history of deep learning and he talks about the relationship between connectionism and computationalism
* Connectionism refers to how people relate their representations of things
* Computationalism refers to how people logically break down situations
* A famous example to demonstrate this puts a cat inside of a closed box. There’s a lever inside of the box that opens a door. Connectionists would argue that through enough trial and error, eventually the cat would learn that pressing the lever opens the door. The cat learned through its experiences and formed a relationship. Computationalists would argue that until the cat explicitly understands the lever opens the door, it’s unable to get out of the box. In other words, it needs to logically make that connection to learn anything.
* This relates to deep learning by thinking about how people actually learn and the different approaches that can be taken to achieve a successful learning model.
* One of the first examples of an AI being able to teach itself came from Bell Labs in 1989 and used relatively modern techniques to be able to decipher handwritten text.
* I’ll get into these techniques here in just a moment, but the number of applications for learning models really is limitless
* As of the mid 2010’s, some of the most popular areas of study for this topic dive into healthcare, transportation, speech recognition, music composition, robotics, art, and even earthquake predictions

// Overview of content //

* To oversimplify how deep learning occurs, one of the ways AI models are able to train themselves is by taking in input data and using it to correctly assess new data that’s given to it.
* So here I’ve created an illustration to help break down this process and explain some of the technicalities in understanding the theory behind this
* \*Show diagram
* To start, the architecture of a deep learning model consists of input layers, output layers, and hidden layers in between
* The data that’s inputted is broken down into smaller parts that can be represented as neurons
* These neurons are like neurons in the brain that fire when a connection is made or a pattern is recognized
* A really popular example to illustrate deep learning is through a number detection system.
* This particular AI allows the user to draw a number 1 through 9 and tells the user which number they drew
* The amount of shading for each pixel is represented as a number between 0 and 1, with 1 being a fully shaded pixel. Each neuron within the first layer represents one pixel and holds the shading value
* This data is added together and passed through the neurons in the middle layers using a nonlinear function that assigns a weight and bias to each connection
* \*Show equation
* The weight and bias of a neuron represent its importance in identifying a particular pattern in the data
* For example, in identifying numbers drawn, certain groups of pixels in a particular region could represent the edges present in specific numbers
* This means this region will theoretically have a greater weight
* Breaking the input down into edges helps identify the purpose behind one of the middle layers
* Breaking this down even further, the edges can be combined to identify loops or longer edges
* If enough pixels in this edge are shaded in, the neuron within the next layer will be very active. \*Show horizontal middle edge
* Combining this with other layers will cause the next layer to make a decision and eventually produce a final result
* \*AI is a bunch of if statements meme
* This might make AI sound like a bunch of if statements
* However, it’d be extremely tedious to manually set the weights and biases for every neuron so that the system could correctly identify every pattern initially. You’re dealing with potentially tens of thousands of alterations for a relatively simple algorithm such as this one.
* This is where the different techniques for training AI come in
* Two main processes define the main technique used and they’re called forward and backward propagation
* Forward propagation refers to the process that we’ve just discussed.
* Input data is fed through several weighted middle layers to produce a decision that’s outputted
* The data moves forward and you can see how this continues on through the rest of the graph
* Taking a look at the weight equation from before, inputs are processed through what’s called an activation function that condenses the data into a more useful number that represents how active a neuron should be \*List activation functions on slide
* If I add every weighted input together I’m going to get a large number, so the activation function essentially converts this information back into a number between 0 and 1
* There are several different functions you can use in a neural network, but this ties into the main goal of forward propagation which is to take the weighted sum of each of the inputs from the previous layer, add on a bias to account for additional importance and run it through an activation function to achieve a neuron’s activity
* This process is repeated for every neuron as it progresses through the graph
* Initially, as previously mentioned, you wouldn’ want to manually set the weights and biases for each connection since it would take way too long to determine the ideal values
* This also relates to why the middle layers are referred to as hidden; we have no idea how they should look at first
* Backwards propagation utilizes calculus and linear algebra to take the output and alter the weights and biases of the neurons so that the system can get a clearer idea of what a correct output should look like
* One of the main functions used in this process is called a cost function and helps define why deep learning needs lots of data to flourish
* Let’s say that starting off we randomly set the weights and biases for each connection so that we have something to work with
* As you could imagine, an AI model processing data for the first time using this system isn’t going to be very accurate and the output is going to look very chaotic
* The cost function takes the *desired* output and uses it to give the network an idea for how accurate it is
* Say that we’re given this set of output data
* The cost function takes the summation of the square of the differences between the desired and actual outputs, and it looks like this \*Show cost function being used
* This gives you a value that represents the cost of the network, or how accurate it was to matching that one example
* Since you want the network to output what matches the dataset, you subtract the most from the correct output (1)
* Applying this to an ideal output would yield 0 cost since 1 - 1 is 0, and so is 0 - 0
* A network that yields a very high cost is inaccurate, while one that yields a low cost is more accurate
* Taking this a step further, if we were to graph this data for every example in our dataset, we’d have a function that takes every weight and bias as an input, and the cost as an output
* This means there exists a set of weights and biases that yield the smallest possible cost
* Using calculus, we can use this graph to calculate how we need to change the network so that we can minimize this cost
* This can be compared to finding the local minimum of a graph, which is much more conceptually easy to understand
* A global minimum for this graph would be considered perfect, but often we have to settle for local minima that are still fully capable of being practically useful
* If we were to start at a given point, there exists a direction that we can move so that the costs decreases the fastest
* This process is called gradient descent and essentially tells us how we can change the weights and biases so that we can figure out the lowest cost the fastest
* I don’t understand multivariable calculus and I’m not sure how many of you are there with me, but all that matters for this discussion is that this process tells us how we can change each weight and bias for the previous layer to push the network one step closer to where it needs to be
* If this process is repeated for each neuron in each layer, you’re essentially moving backwards through the graph to adjust how it behaves
* So just to recap, initially, we have no idea what the network needs to be set to and we start with random weights and biases. Next, we take the random output and compare it to the desired output from our data and calculate a cost function to determine how accurate its assessment was. Then, we use that cost data to calculate the gradient descent to tell us how we need to adjust the weights and biases for the previous layer to achieve a more accurate solution
* The more data that we feed into the network, the more accurate it becomes, because each instance of data pushes us closer to the minimum cost using this method

// Drawbacks / More general info //

* So what are the drawbacks of deep learning?
* One common problem that deep learning architectures run into involves overfitting, where the network performs very well when presented with training data, but very poorly when given new unseen data
* On the other side of the token, networks can also be underfitted and perform poorly for all types of data depending on how they’re structured
* This can be caused by adding too many middle layers
* More hidden layers doesn’t equate to a more accurate algorithm, it just prevents backwards propagation from being as effective towards the initial layers and increases the time complexity
* Making this as efficient as possible could lead into a new section of research

// Where is AI headed? //

* So what does the future of AI look like?
* One area that I’d like to look more into involves taking a deep learning model and applying it to several different machines at the same time
* This is called transfer learning and I’ve found that there’s been very little research done on this topic
* Essentially, if I were to set up two of the same deep learning models on two separate machines and have them run the same data in a different order, I’d end up with similarly accurate networks that were able to achieve their accuracy independent of one another
* Transfer learning takes this a step further and derives the results from one machine to teach the second machine how to recognize similar patterns
* Jeremy Howard, a professor and the founder of a company called fast.ai, spoke in an interview about how he developed a language processor using transfer learning and how it outperformed most state of the art language processors in a fraction of the amount of time with less data
* This can be compared to riding a bike for the first time
* The first time you get on a bike, you have to learn how to balance yourself, steer, brake
* However, once you master riding a bike, riding a motorcycle for the first time is much easier because you already understand several of the fundamentals that you’ve learned previously
* Theoretically, the independence that machines would have from one another would be removed and they’d be able to continue building off of eachother
* As Jeremy spoke about, this can significantly cut down on input data and time
* However, in a study done by Jean-Pierre Briot, where he developed a music composition AI, he found that combining networks didn’t always yield better results
* Rather, he said that combining the right ones was what mattered
* He compared it to the way a chef doesn’t combine every ingredient in a recipe and instead picks the right ones

// Conclusion //

* In conclusion, recent advances in deep learning hold a promising future for AI and open up many different possibilities for practical solutions
* Hopefully I was able to explain that relatively well. If you have any questions about some of the theory behind deep learning, I referenced this book by Michael Nielsen called “Neural Networks and Deep Learning” where he talks about how you can get started setting up a neural network and incorporate all of the math behind deep learning into an algorithm.
* Here were some of the other papers I pulled from if you wanted to review any of the other studies that were conducted
* Do you guys have any questions for me?