FUNDAMENTALS OF TinyML [outcomes]

* Fundamentals of Machine Learning (ML)
* Fundamentals of Deep Learning
* How to gather data for ML
* How to train and deploy ML models
* Understanding embedded ML
* Responsible AI Design

**TinyML will soon be everywhere:**

* TinyML will soon be everywhere, powering the next generation of smart embedded devices. These devices will be in our homes and in very remote locations, enabling remote monitoring for both industry and ecology. Today, in these remote monitoring settings, 99% of raw sensor data is discarded, which is a wealth of data for machine learning!
* TinyML can provide a unique solution: by summarizing and analyzing data at the edge on low power embedded devices, TinyML can provide **smart** summary statistics that take these previously lost patterns, anomalies, and advanced analytics into account [[1](https://www.mckinsey.com/~/media/McKinsey/Industries/Technology%20Media%20and%20Telecommunications/High%20Tech/Our%20Insights/The%20Internet%20of%20Things%20The%20value%20of%20digitizing%20the%20physical%20world/The-Internet-of-things-Mapping-the-value-beyond-the-hype.pdf)] [[2](https://hbr.org/webinar/2017/04/whats-your-data-strategy)].
* In this reading, we survey a few emerging application areas that have great potential for TinyML. This list is a tiny (no pun intended) preview into the wealth of applications on the horizon. Later in this course we will do some basic review of machine learning, which some of you will need less than others but it is still a good review. In the next two courses we will be diving in much deeper around TinyML.

**Industrial Predictive Maintenance:**

* In the industrial setting, TinyML is already being used to provide smarter sensing that enables advanced monitoring improving productivity and safety. For example, maintenance and monitoring of remote wind turbines can be quite challenging and time-consuming. However, if we could proactively predict that the machine will have trouble, we can predictively do maintenance ahead of any failures. Such “predictive maintenance” can lead to significant cost savings due to reduced downtimes, better availability of the systems for higher reliability in the product, which leads to overall higher quality of service for end-users/customers.
* There are many TinyML applications for predictive maintenance. For instance, an Australian startup, Ping Services, has introduced a novel IoT device that continuously and autonomously inspects a turbine as it’s running. By magnetically attaching to the outside of any turbine (notice the small device in the image below) and analyzing detailed data at the edge and summary data in the cloud, the device can efficiently and effectively alert of any potential issues before a problem arises inside the turbine [[3](https://ping.services/)].

#### Agriculture:

Every day, the cassava crop provides food for more than 500 million African people. However, this vital stable is continuously under attack from a variety of diseases. The team at PlantVillage, led by Dr. Amanda Ramcharan, has developed the Nuru app to help farmers identify and treat these diseases. By running machine learning using TensorFlow Lite on mobile phones, the app enables real-time mitigation without the need for access to the internet -- a crucial requirement for many remote farmers (see the image below for the system in action). The next generation of this system will go farther -- leveraging tinyML and technologies like TensorFlow for Microcontrollers to deploy sensors across remote farms to enable better tracking and analysis [[4](https://grow.google/intl/europe/story/transforming-farmers%E2%80%99-lives-with-just-a-mobile-phone)].

#### Healthcare:

The Solar Scare Mosquito project deploys small smart Internet of Things (IoT) robotic platforms to help curb the spread of mosquito-borne epidemics such as Malaria, Dengue, and the Zika Virus. The system works by disrupting the mosquito breeding cycle by agitating water likely to contain mosquito larvae. The system uses rain and acoustic sensors to determine when it needs to agitate water to conserve battery and enable it to run on solar power indefinitely. It also sends smart summary statistics and alerts to warn of possible mosquito mass breeding events over lower power low-speed communication protocols. By making the system self-sufficient, small, and affordable, these devices can be deployed widely, preventing mosquitoes spread. All of the necessary components are included in a single component smaller than the size of a soccer ball [[5](https://hackaday.io/project/174575-solar-scare-mosquito-20)].

#### Wildlife Conservation:

On the Land:

TinyML is also already being used for ecological and environmental monitoring. For example, over the past 10 years, the Siliguri-Jalapaiguri railway line in India has had over 200 fatal collisions with elephants. Researchers from the Laboratory of Applied Bioacoustics at the Polytechnic University of Catalonia designed a smart acoustic and thermal sensor system using custom machine learning models running on solar power as an early warning system (see image below—all-in-one package with self-sustained energy source allows proximity to railway without added infrastructure, e.g., power lines) [[6](https://www.zdnet.com/article/elephants-vs-trains-this-is-how-ai-helps-ensure-they-dont-collide/)].

And In the Sea:

Similar systems are also being deployed in the waterways around Seattle and Vancouver to prevent whale strikes in busy shipping lanes. These smart ML powered sensors enable constant real time monitoring and increased density of sensor deployments improving overall system efficiency and efficacy [[7](https://graphics.wsj.com/glider/google-builds-ai-to-help-ships-and-whales-coexist-f4b74f53-bba5-442f-90a4-e42dfb13dad2?mod=e2twd)].

**Responsible AI**

The following three case studies are all real-world examples of when AI has failed in some way. First, read through the following descriptions of each case:

1. Winterlight Labs auditory detection of Alzheimer’s disease

In 2016, Winterlight labs designed an AI-powered auditory test for Alzheimer’s disease, where users’ speech would be recorded and AI would be used to detect signs of Alzheimer’s such as vocabulary richness, pauses in speech, and syntactic complexity. However, the initial research findings revealed a serious problem; non-native English speakers were being inaccurately flagged as having Alzheimer’s disease. Since the data that was used to train the model had been collected from native English speakers from Ontario, Canada, this technology was unable to work reliably across different populations.

2. Wireless baby monitors hacked

In 2018, there were several instances where wireless baby monitors were hacked which ultimately made national news headlines. In one case, a hacker used his newfound access to the baby monitor device to broadcast threats and shout sexual expletives. In another case, a more benevolent hacker used his newfound access to warn parents about the susceptibility of their device, in hopes that the parents would be able to address the situation before being targeted by nefarious hackers.

3. Hidden microphones in Nest devices

In 2019, users of Nest Guard devices were shocked to discover hidden microphones inside the device. Users were concerned about the invasion of privacy as well as the breach of trust that resulted from not being properly informed about the specs of the device. From Google’s perspective, the on-device microphone was simply a form of future-proofing that would allow the device to be compatible with updates that supported new functions later down the line.

Now, reflect on the following questions and join the discussion going on in the forum!

A. Which one of these cases do you find most concerning? Which concerns you the least?  
B. What do you consider to be the relevant ethical challenges?   
C. What do you think the designers of this technology could have done differently?  
D. How can you apply learnings from these examples to your own job? Your personal life?  
E. Do you agree or disagree with what others have already posted in the forum? Why?

**Finding patterns**

In the previous video you learned about how traditional programming is where you explicitly figure out the rules that act on some data to give an answer like this:



And then you saw that Machine Learning changes this, for scenarios where you may not be able to figure out the rules feasibly, and instead have a computer figure out what they are. That made the diagram look like this:



Then you read about the steps that a computer takes -- where it makes a guess, then looks at the data to figure out how accurate the guess was, and then makes another guess and so on.

So, consider if I give you a set of numbers like this:

*X: -1, 0, 1, 2, 3, 4*

And then I give you another set of numbers like this:

*Y: -3, -1, 1, 3, 5, 7*

Can you figure out the relationship between the two sets? There’s a function that converts -1 to -3, 0 to -1, 1 to 1, 2 to 3, 3 to 5 and 4 to 7. Can you figure out the relationship. Think about it for a moment.

Often when I ask people about it, they see that the 0 is matched to -1, so Y is (something) times X - 1. Maybe they’ll take a guess at the something, and come up with 3.

Then fill in the gaps, if Y=3X-1, then

X: -1, 0, 1, 2, 3, 4

Becomes

Y: -4, -1, 2, 5, 8, 11

Other than working for 0, it fails for everything else. In ML terms, you can define this as your loss is ‘high’.  With what you learned from that, you might think, what if it’s Y=2X-1?

Then, when you fill in the results for Y=2X-1, you’ll get:

*X: -1, 0, 1, 2, 3, 4*

Becomes

*Y: -3, -1, 1, 3, 5, 7*

...which matches your original data perfectly. Your loss is zero.

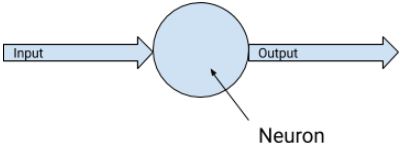
You’ve just gone through this process:



# **More neural networks**

Up to now you’ve been looking at matching X values to Y values when there’s a linear relationship between them. So, for example, you matched the X values in this set [-1, 0 , 1, 2, 3, 4] to the Y values in this set [-3, -1, 1, 3, 5, 7] by figuring out the equation Y=2X-1.

You then saw how a very simple neural network with a single neuron within it could be used for this.



This worked very well, because, in reality, what is referred to as a ‘neuron’ here is simply a function that has two learnable parameters, called a ‘weight’ and a ‘bias’, where, the output of the neuron will be:

Output = (Weight \* Input) + Bias

So, for learning the linear relationship between our Xs and Ys, this maps perfectly, where we want the weight to be learned as ‘2’, and the bias as ‘-1’. In the code you saw this happening.

When multiple neurons work together in layers, the learned weights and biases across these layers can then have the effect of letting the neural network learn more complex patterns. You’ll learn more about how this works later in the course.

In your first Neural Network you saw neurons that were densely connected to each other, so you saw the **Dense** layer type. As well as neurons like this, there are also additional layer types in TensorFlow that you’ll encounter. Here’s just a few of them:

* **Convolutional** layers contain filters that can be used to transform data. The values of these filters will be learned in the same way as the parameters in the Dense neuron you saw here. Thus, a network containing them can learn how to transform data effectively. This is especially useful in Computer Vision, which you’ll see later in this course. We’ll even use these convolutional layers that are typically used for vision models to do speech detection! Are you wondering how or why? Stay tuned!
* **Recurrent** layers learn about the relationships between pieces of data in a sequence. There are many types of recurrent layer, with a popular one called LSTM (Long, Short Term Memory), being particularly effective. Recurrent layers are useful for predicting sequence data (like the weather), or understanding text.

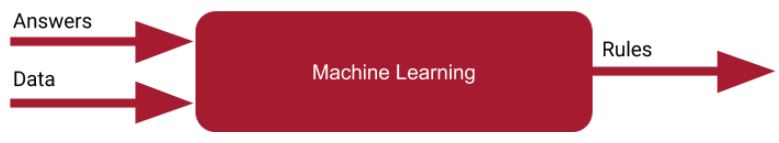
You’ll also encounter layer types that don’t learn parameters themselves, but which can affect the other layers. These include layers like **dropouts**, which are used to reduce the density of connection between dense layers to make them more efficient, **pooling** which can be used to reduce the amount of data flowing through the network to remove unnecessary information, and **lambda** lambda layers that allow you to execute arbitrary code.

# **Neural networks in action**

You’ve now seen a very simple example for how computers can learn. There’s no great mystery to it -- it’s a simple algorithm of making a guess, measuring how good that guess is (aka the loss), and then using this information to optimize the guess, and continually repeating this process to improve the guess.

# **Initialization and learning**

Recall that Machine Learning, the core process powering TinyML, is the process of having Data and Answers, and from them attempting to infer the rules that link them:



So, if your **data** is the numbers in this set: [-1, 0 , 1, 2, 3, 4] , and your corresponding **answers** are the numbers in this set: [-3, -1, 1, 3, 5, 7], one way to begin to infer the relationship between them (assuming it’s linear) is to have a function, where we say Y = wX+b, and we have to try to figure out the values of w and b.

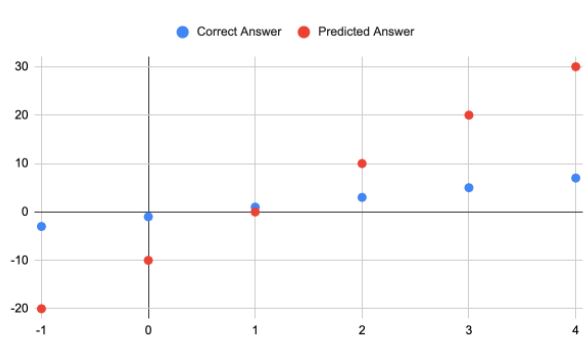
A process to do this is shown below:



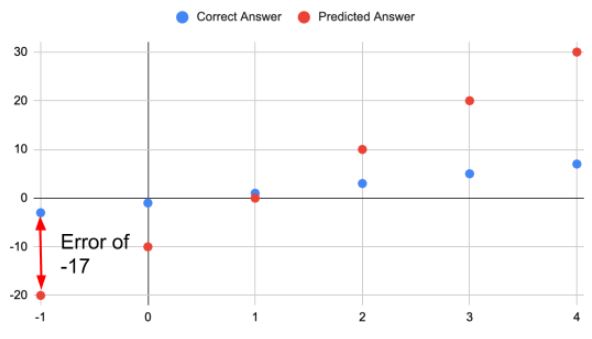
We could make a guess as to the values of w and b, and measure how accurate that guess is.

So, for example, if we guess that w=10, and b = -10, we would get the set of **answers** for our data as [-20, -10, 0, 10, 20, 30], where our *correct* answers are [-3, -1, 1, 3, 5, 7].

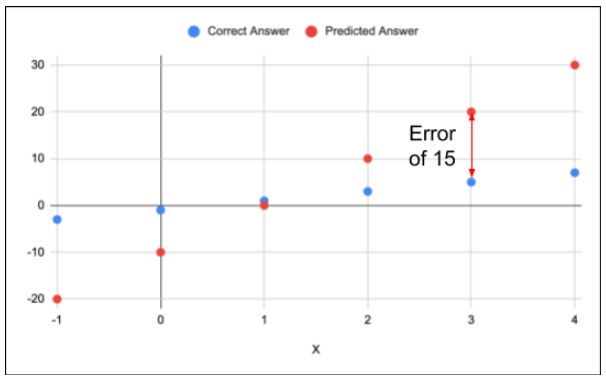
From this we can measure our loss, by plotting our predicted answer against our actual answer.



So, for example, for x=-1, we predicted -20, when the answer is actually -3, so we are off by -17



And when x=3, we predicted the answer to be 20, when in fact it is supposed to be 5, so we have an error of 15.

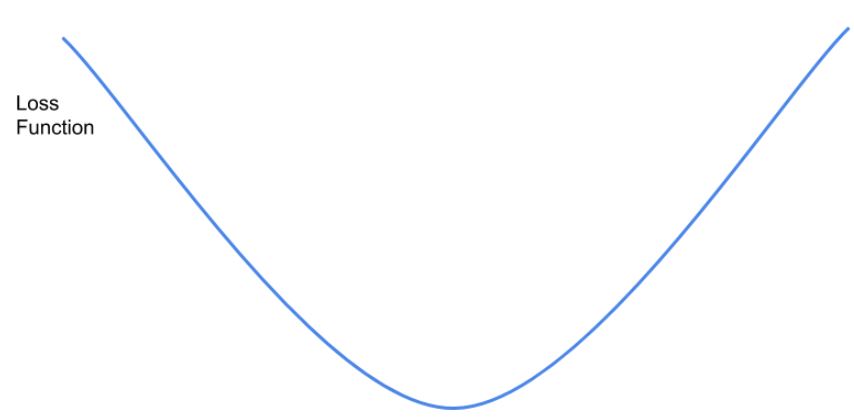


If we want to find out our error, we could average each of these.

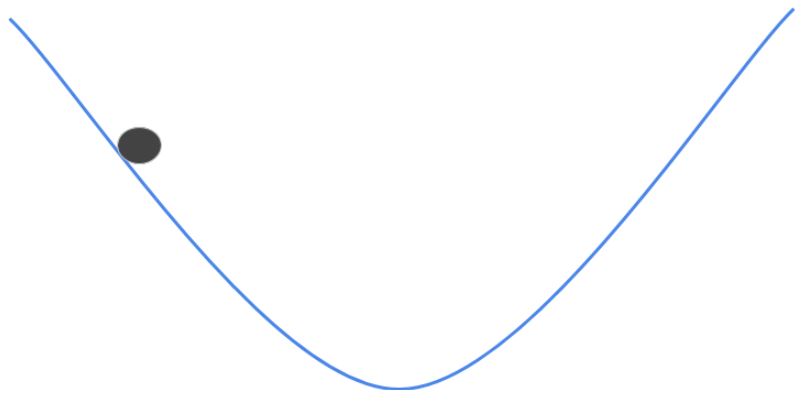
However, by doing this our negative (-17) and positive (15) errors would mostly cancel out, so the smart thing to do to ensure that we don’t have them do this is to square the error amount, average out the squares of the errors, and then get the square root of that.

This gives us a loss function of root mean squared error.

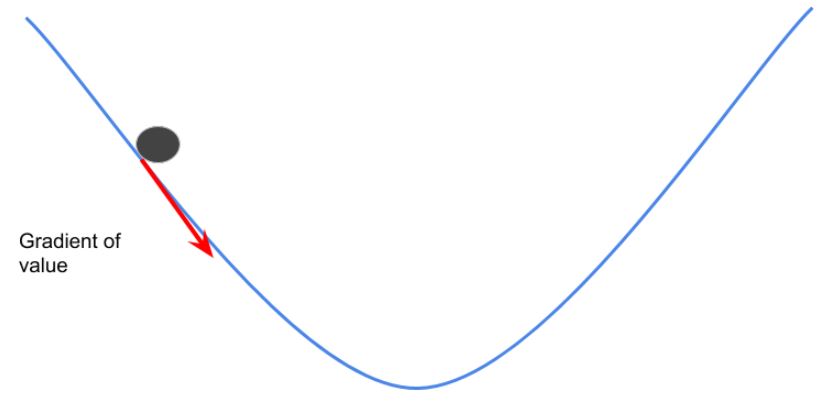
When we now want to optimize our guess, and do better than these results, we can look at the loss function, and figure out a way to minimize our loss. As the loss function uses a square, the curve of the function is parabolic.



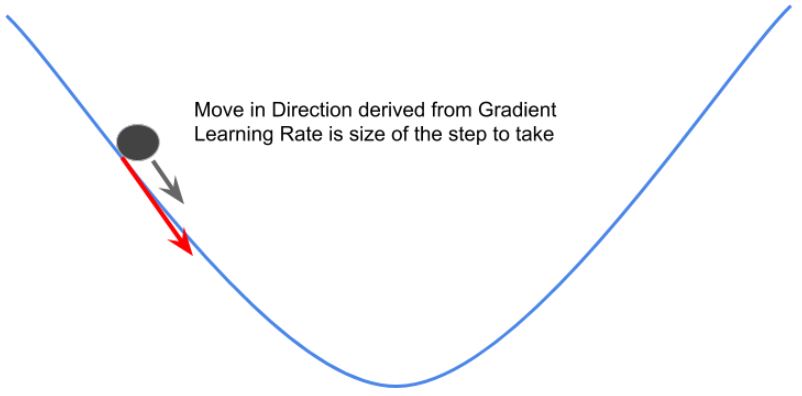
And the value of our loss with our current parameters can be plotted on this. Our loss was high, so we can say we’re pretty far up the curve.



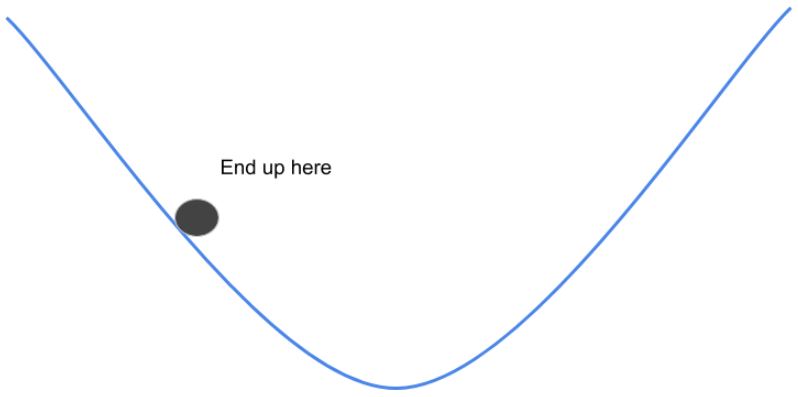
When we differentiate our values against the loss function, we’ll get a gradient, and we can use the gradient to figure out which direction we have to go in to move down the slope!



And we can then move down the slope in the direction derived from gradient. We’ll ‘jump’ by a certain amount, and we can call that amount the ‘Learning Rate’



After that, we are now closer to the bottom of the curve. The parameters that give us our location on the curve can then be the next ‘guess’, and by definition, these will have a lower loss, and we can repeat the loop. This is called back propagation.



The process can be repeated, and over time the value will get closer and closer to the bottom of the parabola, which is the minimum value of the loss function.

This process of using the gradient of the value to figure out the direction of the minimum, and then jumping down the curve towards the minimum is called gradient descent.