In the **Machine Learning with Python** course, you will learn about applied machine learning using Python tools, which is essential for careers in data science and AI. Key points include:

* **Python's Dominance**: Python is the most widely used language in machine learning.
* **Course Structure**: The course covers the machine learning lifecycle, modeling techniques (classification, regression, clustering), and advanced methods like reinforcement learning and deep learning.
* **Hands-On Experience**: You will build, assess, and validate machine learning models using libraries such as Pandas, NumPy, and Scikit-Learn.
* **Learning Objectives**: Gain foundational skills in machine learning, data preparation, and model evaluation.
* **Final Project**: Apply your knowledge in a concluding project.

**summary of the current course content:**

* **Course Overview**: This course is part of the IBM Professional Certificate programs in AI Engineering and Data Science.
* **IBM Data Science PC**:
  + Focuses on data science skills.
  + Covers data cleaning, analysis, and predictive modeling using tools like Python, SQL, Pandas, and NumPy.
  + Designed for beginners with no prerequisites.
* **IBM AI Engineering PC**:
  + Prepares learners for AI engineering roles.
  + Topics include building, training, and deploying models, including large language models (LLMs).
  + Requires knowledge of Python and data analysis basics.
* **Key Skills Covered**:
  + Machine learning algorithms (e.g., linear regression, decision trees).
  + Deep learning techniques using libraries like Keras and TensorFlow.
  + Generative AI models and natural language processing (NLP).
* **Hands-on Learning**: Both programs emphasize practical projects and labs to build real-world experience.
* **Capstone Project**: Apply deep learning skills to a real-world problem, showcasing your expertise.

**Overview of Machine Learning**

* **Machine Learning (ML)** as a subset of **Artificial Intelligence (AI)** that enables computers to learn from data and make decisions without explicit instructions.
* Different **machine learning techniques**:
  + **Supervised Learning**: Uses labeled data to make predictions.
  + **Unsupervised Learning**: Finds patterns in unlabeled data.
  + **Semi-Supervised Learning**: Combines labeled and unlabeled data.
  + **Reinforcement Learning**: Learns through interaction with the environment.
* Key **techniques** include:
  + **Classification**: Predicts categories (e.g., benign vs. malignant cells).
  + **Regression**: Predicts continuous values (e.g., house prices).
  + **Clustering**: Groups similar data points (e.g., customer segmentation).
  + **Anomaly Detection**: Identifies unusual cases (e.g., fraud detection).
  + **Recommendation Systems**: Suggests items based on user preferences.
* **Applications** of machine learning in real-world scenarios, such as predicting diseases, analyzing consumer behavior, and recognizing images.

**Machine Learning Model Lifecycle**

key processes involved in developing a machine learning product. The main steps include:

* **Problem Definition**: Clearly stating the problem or situation to be addressed.
* **Data Collection**: Gathering relevant data from various sources.
* **Data Preparation**: Cleaning and transforming the data, often referred to as the ETL (Extract, Transform, Load) process.
* **Model Development and Evaluation**: Building the model and assessing its performance.
* **Model Deployment**: Implementing the model in a production environment.

It's important to note that the model lifecycle is **iterative**, meaning you may need to revisit earlier steps based on findings during evaluation or deployment.

**Data Scientist vs AI Engineer**

* **Roles**:
  + **Data Scientist**: Acts as a data storyteller, using mathematical models to analyze and derive insights from structured data.
  + **AI Engineer**: Functions as an AI system builder, utilizing foundation models to create generative AI systems that transform business processes.
* **Use Cases**:
  + Data scientists primarily engage in descriptive and predictive analytics, using techniques like exploratory data analysis and machine learning models (e.g., regression and classification).
  + AI engineers focus on prescriptive use cases (e.g., decision optimization) and generative use cases (e.g., creating intelligent assistants and chatbots).
* **Data Types**:
  + Data scientists often work with structured data, requiring extensive cleaning and preprocessing.
  + AI engineers primarily handle unstructured data (text, images, etc.) and utilize large-scale foundation models.
* **Models**:
  + Data scientists use a variety of models tailored to specific tasks, while AI engineers rely on foundation models that can generalize across multiple tasks.
* **Processes**:
  + The data science process involves selecting data, training models, and deploying them.
  + The AI engineering process often starts with a pre-trained model, leveraging prompt engineering to interact with foundation models.

**Tools for Machine Learning**

* **Machine Learning Tools**: These tools facilitate machine learning pipelines, including data preprocessing, model building, evaluation, and optimization. Examples include:
  + **Pandas**: For data manipulation and analysis.
  + **Scikit-learn**: For supervised and unsupervised learning algorithms.
* **Programming Languages**: Common languages for machine learning include:
  + **Python**: Widely used for its libraries and ease of use.
  + **R**: Popular for statistical learning.
  + Other languages mentioned are Julia, Scala, Java, and JavaScript.
* **Categories of Tools**:
  + **Data Processing and Analytics**: Tools like PostgreSQL, Hadoop, and Spark.
  + **Data Visualization**: Libraries such as Matplotlib, Seaborn, and Tableau.
  + **Machine Learning**: Libraries like NumPy, SciPy, and Scikit-learn.
  + **Deep Learning**: Frameworks like TensorFlow and PyTorch.
  + **Computer Vision**: Tools like OpenCV and Scikit-Image.
  + **NLP Tools**: Libraries such as NLTK and TextBlob.
  + **Generative AI Tools**: Examples include Hugging Face Transformers and DALL-E.

**…………………………………………………………………………………………………**

**Introduction to Deep Learning & Neural Networks with Keras**

**Module 1 - Introduction to Deep Learning**

* Introduction to Deep Learning
* Biological Neural Networks
* Artificial Neural Networks - Forward Propagation

**Module 2 - Artificial Neural Networks**

* Gradient Descent
* Backpropagation
* Vanishing Gradient
* Activation Functions

**Module 3 - Deep Learning Libraries**

* Introduction to Deep Learning Libraries
* Regression Models with Keras
* Classification Models with Keras

**Module 4 - Deep Learning Models**

* Shallow and Deep Neural Networks
* Convolutional Neural Networks
* Recurrent Neural Networks
* Autoencoders

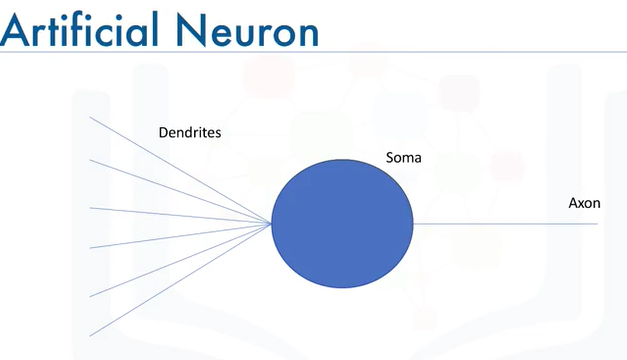
**Introduction to Deep Learning**

* **Deep Learning Popularity**: It is one of the hottest topics in data science, with many exciting projects emerging.
* **Applications**:
  + **Color Restoration**: A system that converts grayscale images to color using convolutional neural networks.
  + **Speech Enactment**: A system that syncs audio clips with video, demonstrated using a speech by Barack Obama.
  + **Automatic Handwriting Generation**: An algorithm that produces realistic cursive handwriting in various styles.
  + Other applications include machine translation, sound addition to silent movies, object classification in images, and self-driving cars.

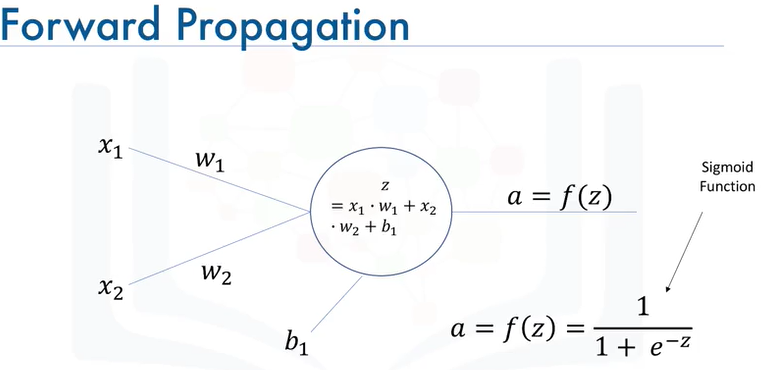
**Neurons and Neural Networks**

* **Neurons**: The structure of a neuron includes the **soma** (cell body), **dendrites** (which receive signals), and **axon** (which transmits signals). The **terminal buttons** at the end of the axon connect to other neurons.
* **Neural Processing**: Neurons process data by combining electrical impulses received through dendrites, which are then sent to the soma for processing and passed along the axon to other neurons.
* **Learning Mechanism**: Learning occurs by strengthening certain neural connections through repeated activation, making them more likely to produce desired outcomes.
* **Artificial Neurons**: Similar to biological neurons, artificial neurons also consist of a soma, dendrites, and an axon, and they learn in a comparable manner.

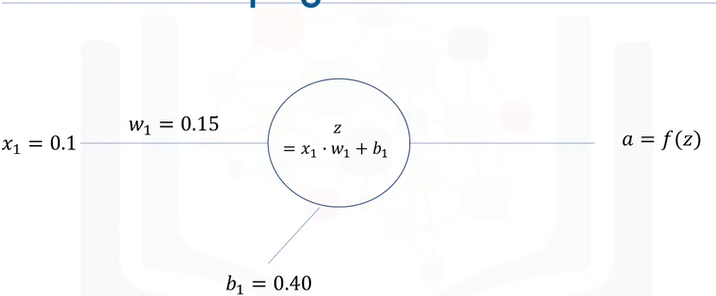
**Artificial Neural Network**

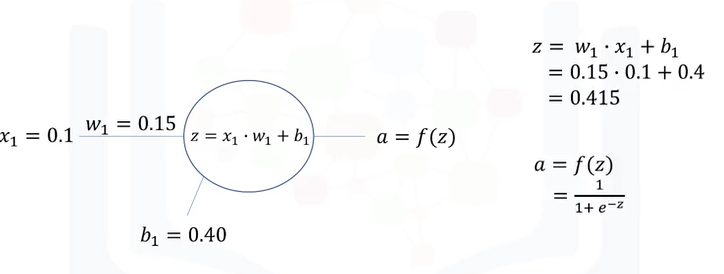


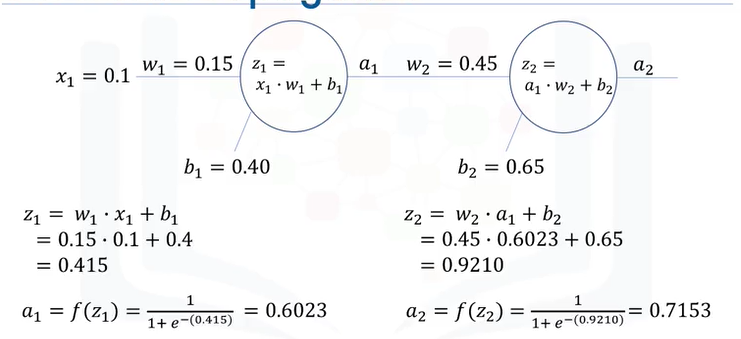
* **Structure of Neural Networks**:
  + **Input Layer**: The first layer that receives input.
  + **Output Layer**: The final layer that produces output.
  + **Hidden Layers**: Layers between input and output that process data.
* **Main Topics**:
  + **Forward Propagation**: The process of data moving through the network from input to output.
  + **Backpropagation**: Not covered in this video but is another key concept.
  + **Activation Functions**: Functions that determine if a neuron should be activated.
* **Forward Propagation Explained**:
  + Data flows through neurons via connections with specific weights.
  + Each neuron computes a **weighted sum** of inputs plus a **bias**.
  + The output is transformed using an **activation function** (e.g., sigmoid function) to introduce non-linearity.
* **Example Calculation**:
  + Given inputs and weights, the output is calculated using the sigmoid function.
  + The process is consistent regardless of network complexity.
* **Importance of Activation Functions**: They enable neural networks to perform complex tasks beyond linear regression.



**Example**

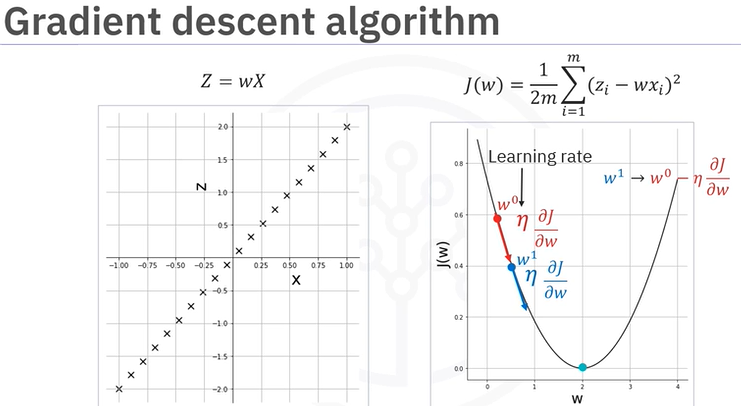
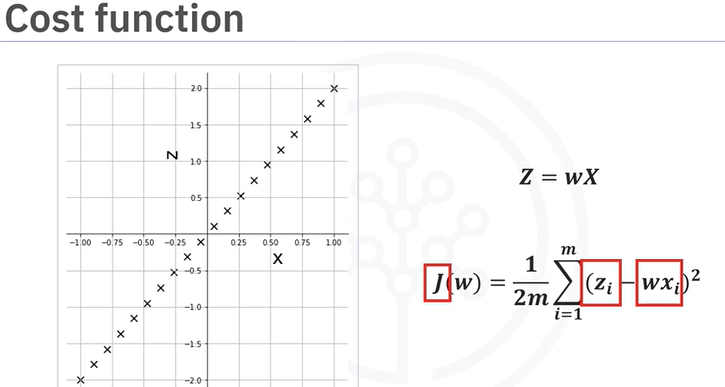
****

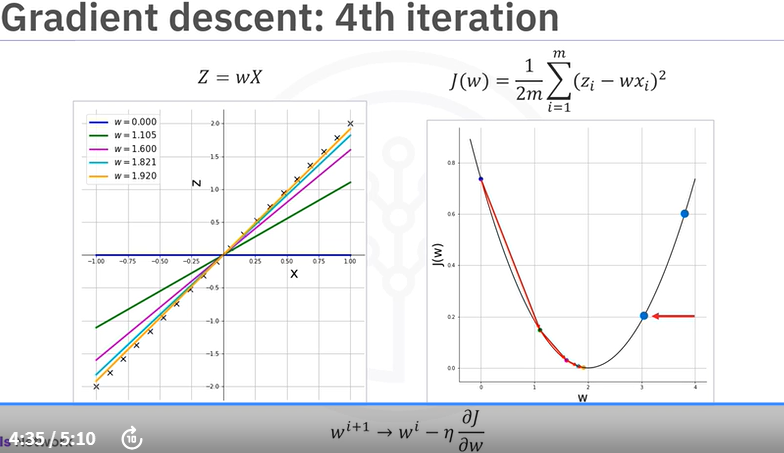
****

****

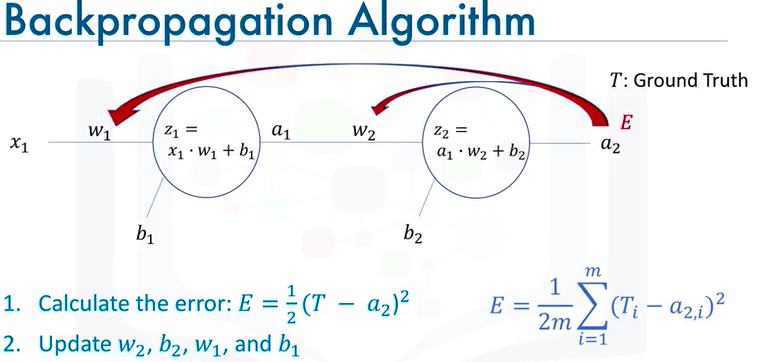
**Gradient Descent**:

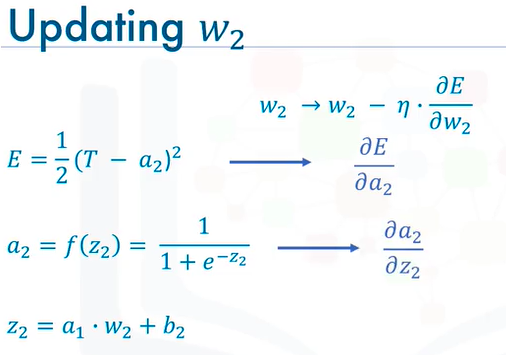
* **Cost Function**: Represented in a scatterplot, it measures how well a model fits the data. A common cost function is defined as the squared difference between actual values (z) and predicted values (w \* x).
* **Finding the Best Value of w**: The goal is to minimize the cost function, which is a parabola with one global minimum. For example, if z = 2x, the optimal value of w is 2.
* **Gradient Descent Algorithm**: This iterative optimization algorithm finds the minimum of a function by taking steps proportional to the negative of the function's gradient at the current point.
* **Learning Rate**: A crucial parameter that controls the size of the steps taken towards the minimum. A large learning rate can overshoot the minimum, while a small one can slow down the convergence.
* **Iterations**: The process is repeated until the minimum is reached or a predefined threshold is met. Each iteration updates the weight based on the gradient.

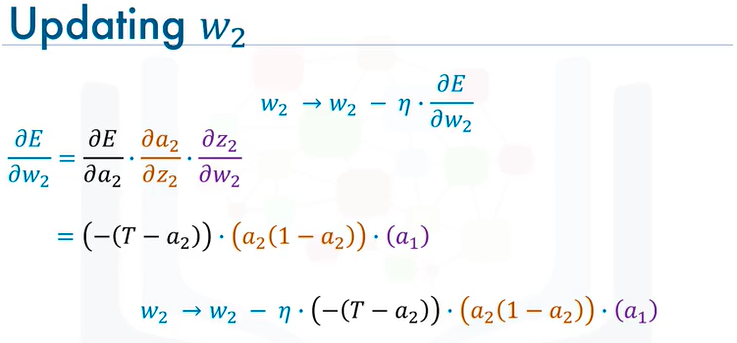
****

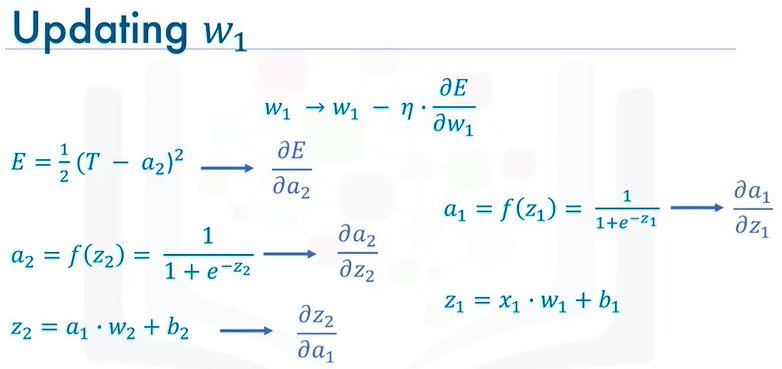
****

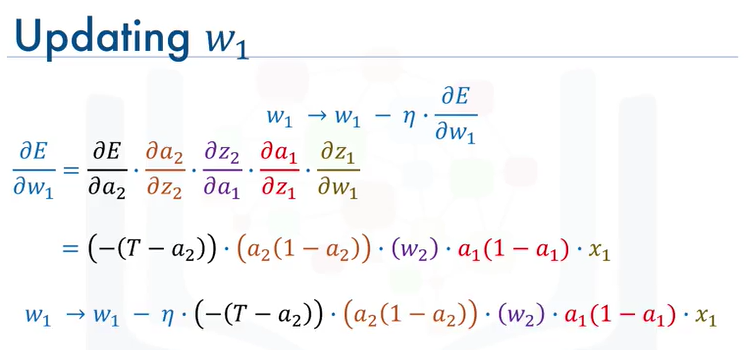
**Backpropagation**

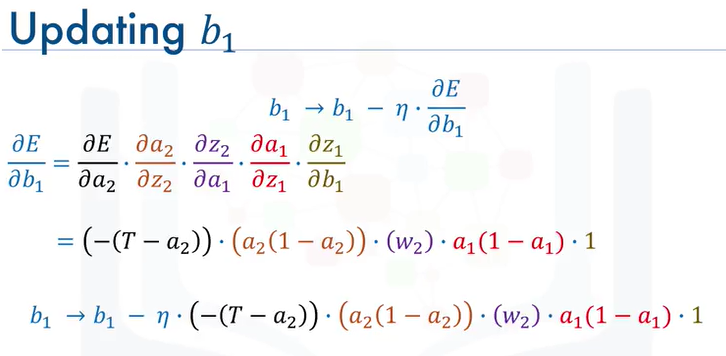
****

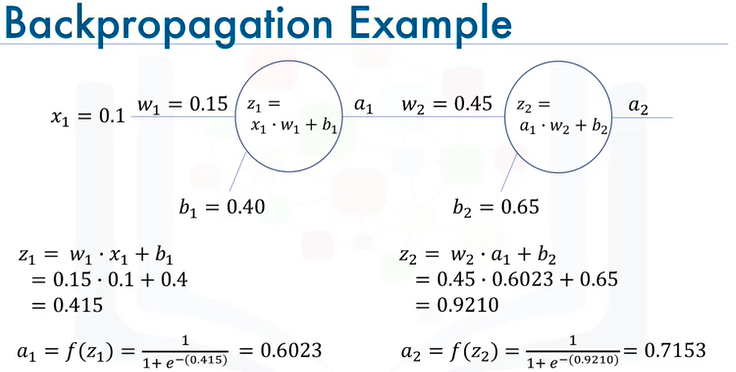
****

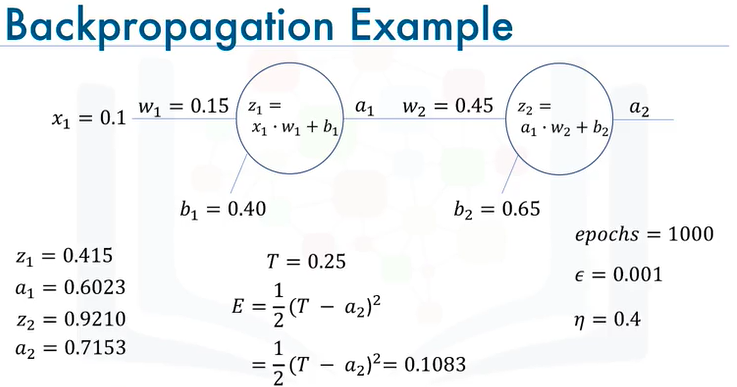
****

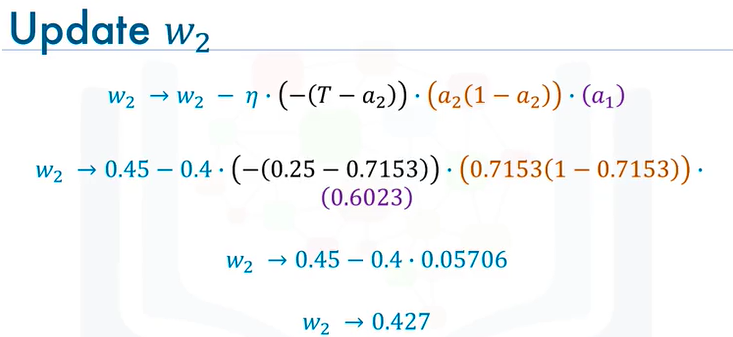
****

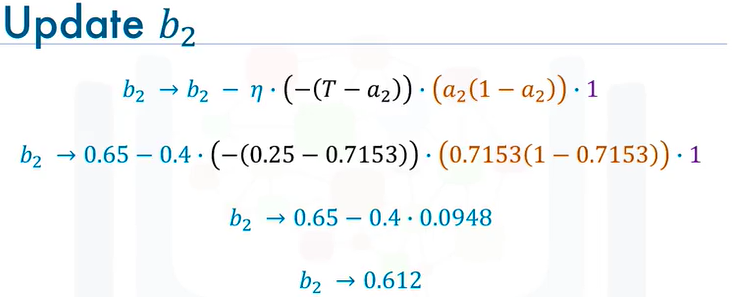
****

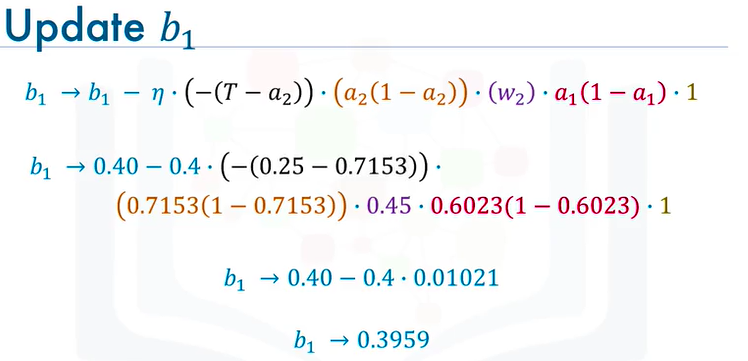
****

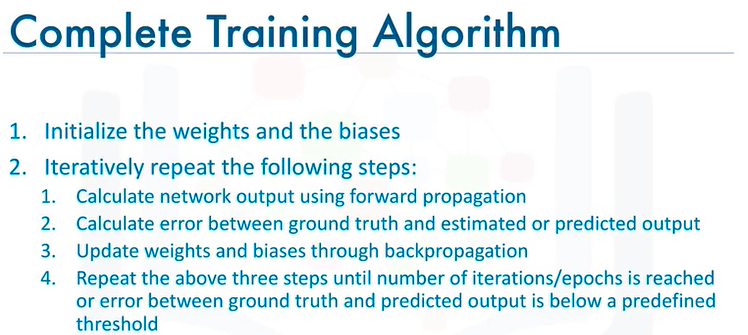
****

****

****

****

****

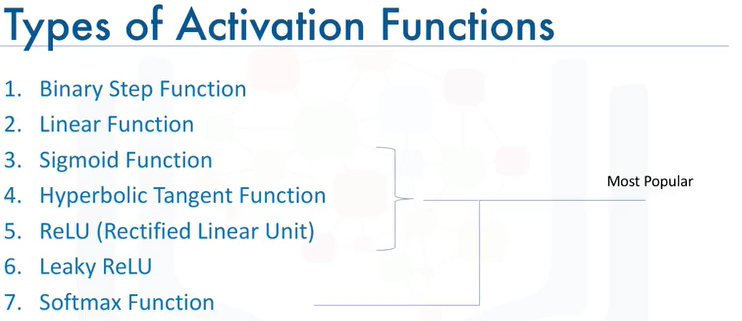
****

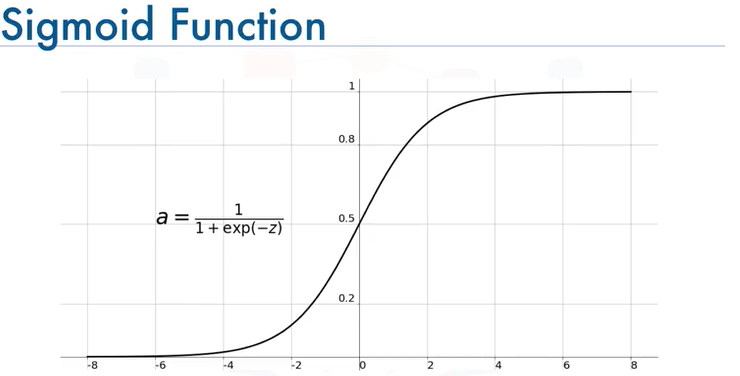
**Vanishing Gradient**

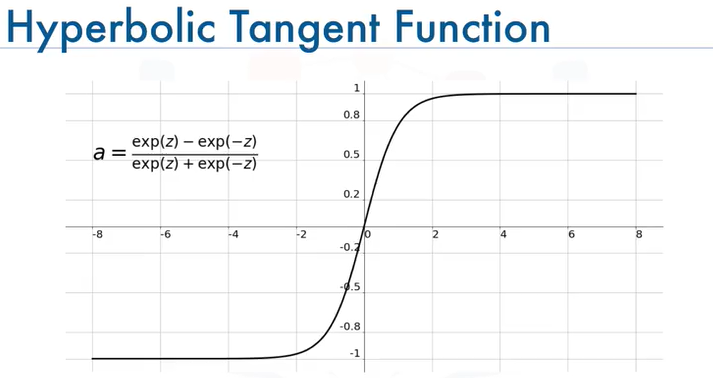
* The sigmoid function restricts intermediate values between 0 and 1.
* During backpropagation, multiplying gradients that are less than 1 leads to increasingly smaller gradients.
* This results in earlier layers of the network learning very slowly compared to later layers, causing a prolonged training process and reduced prediction accuracy.
* Due to this issue, the sigmoid function is generally not used as an activation function in modern neural networks.

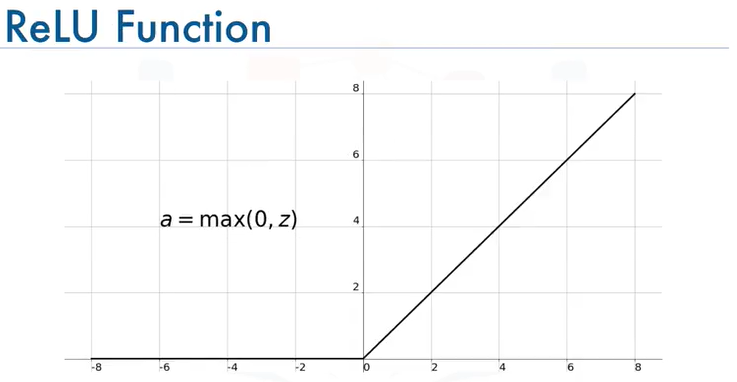
**Activation Functions**

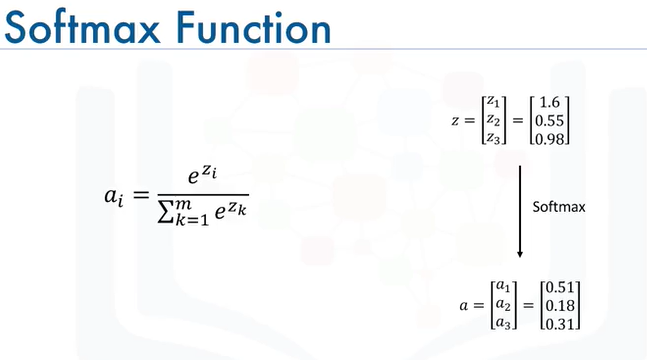
* **Activation Functions Overview**:
  + The sigmoid function has been commonly used but has limitations, such as the **vanishing gradient problem**.
* **Types of Activation Functions**:
  + **Binary Step Function**
  + **Linear (Identity) Function**
  + **Sigmoid Function**: Ranges from 0 to 1, but can lead to vanishing gradients.
  + **Hyperbolic Tangent (tanh) Function**: Ranges from -1 to +1, addressing some sigmoid limitations but still suffers from vanishing gradients.
  + **Rectified Linear Unit (ReLU)**: Most popular today; it activates only a subset of neurons, making it efficient and helping to overcome the vanishing gradient problem.
  + **Leaky ReLU**: A variation of ReLU that allows a small gradient when the input is negative.
  + **Softmax Function**: Used in the output layer for classification tasks, converting outputs into probabilities.

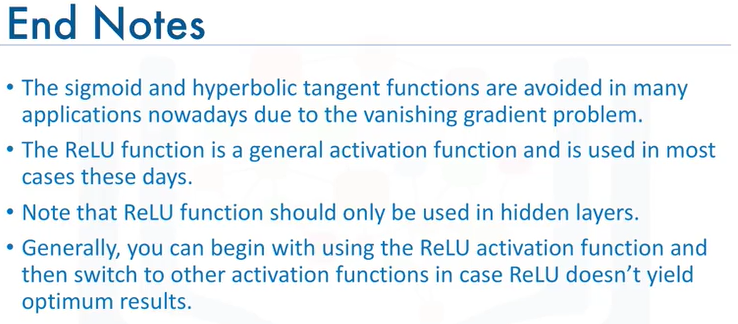
****

****

****

****

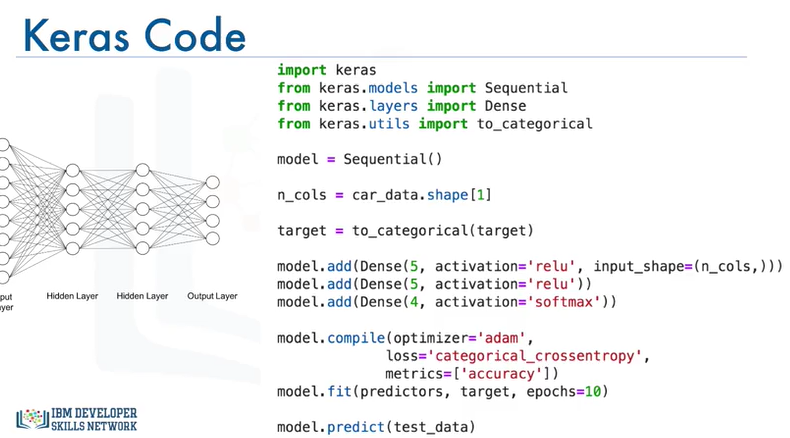
****

****

**Deep Learning Libraries**

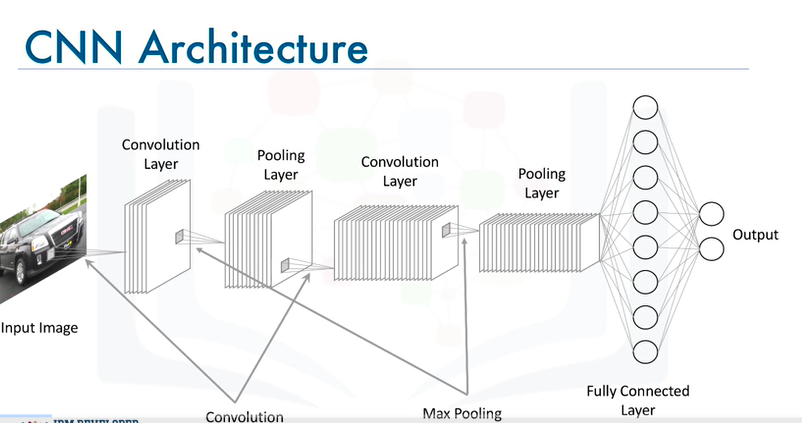
**Regression models with Keras**

* **Model Structure**: A deep neural network is proposed, consisting of:
  + **Input Layer**: 8 features (ingredients of concrete).
  + **Hidden Layers**: Two hidden layers with 5 nodes each.
  + **Output Layer**: One node to predict compressive strength.
* **Dense Network**: All nodes in one layer are connected to all nodes in the next layer.
* **Data Preparation**: The dataset is split into predictors and target columns.
* **Keras Implementation**:
  + Import Keras and the sequential model.
  + Use the add method to build layers, specifying the number of neurons and activation functions (ReLU for hidden layers).
  + Define the optimizer (Adam) and loss measure (mean squared error).
  + Train the model using the fit method and make predictions with the predict method.

****

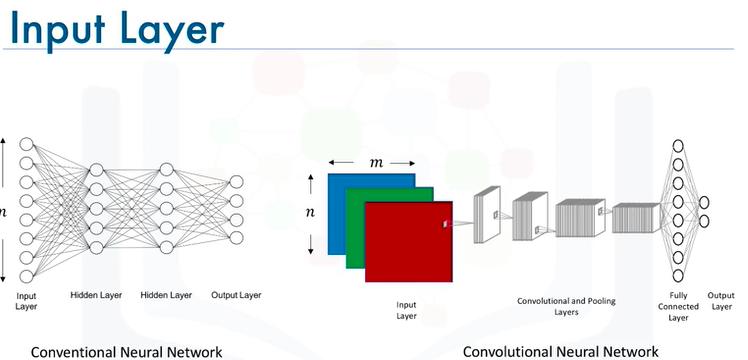
Keras Activation Functions: https://keras.io/activations/ Keras Models: https://keras.io/models/about-keras-models/#about-keras-models Keras Optimizers: https://keras.io/optimizers/ Keras Metrics: <https://keras.io/metrics/>

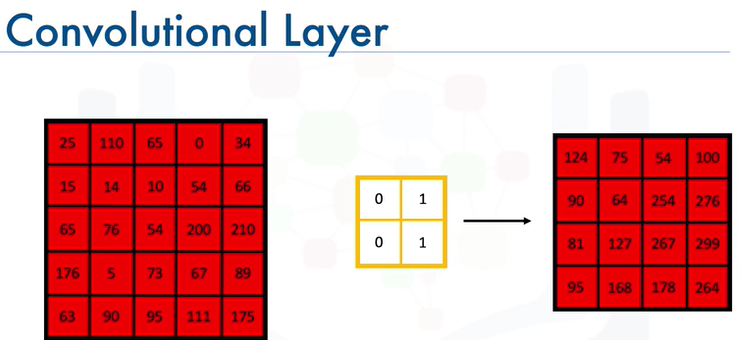
**Convolutional Neural Networks**

****

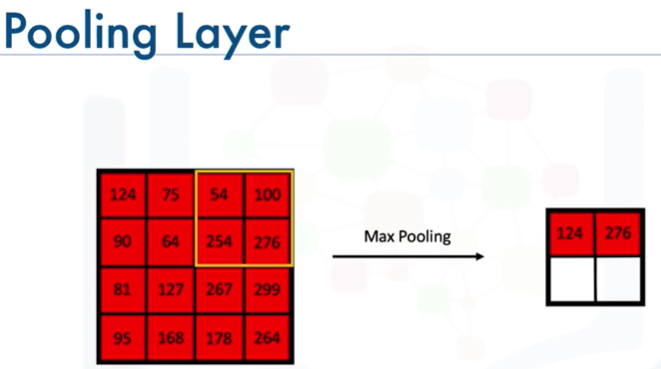
**convolutional neural networks (CNNs)**, a type of supervised deep learning algorithm particularly effective for image-related tasks. Here are the key points:

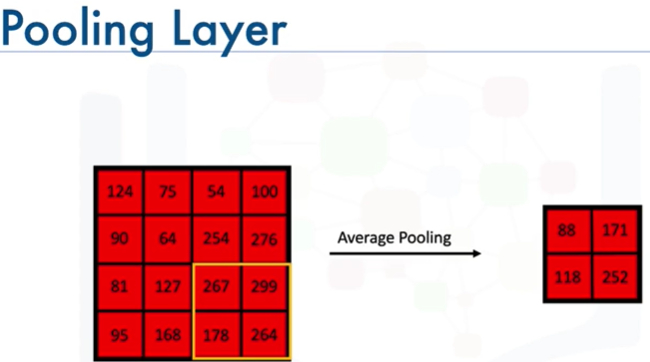
* **Structure of CNNs**: Similar to traditional neural networks, CNNs consist of neurons that optimize weights and biases. However, they are specifically designed for image inputs, which allows for more efficient processing.
* **Input Format**: CNNs accept inputs as (n x m x 1) for grayscale images or (n x m x 3) for colored images, where the three channels represent red, green, and blue.
* **Convolution Layer**: This layer uses filters to perform convolution operations on the input images, preserving spatial dimensions while reducing the number of parameters, which helps prevent overfitting.
* **Activation Function**: The ReLU (Rectified Linear Unit) function is commonly used to filter outputs, allowing only positive values to pass through.
* **Pooling Layer**: This layer reduces the spatial dimensions further, using techniques like max-pooling (keeping the highest value) and average pooling (calculating the average).
* **Fully Connected Layer**: After flattening the output from previous layers, this layer connects every node to the next layer, producing an output vector corresponding to the number of classes in the classification task.
* **Keras Implementation**: The lecture concludes with a brief overview of how to build a CNN using the Keras library, including defining the model, adding layers, and specifying activation functions.

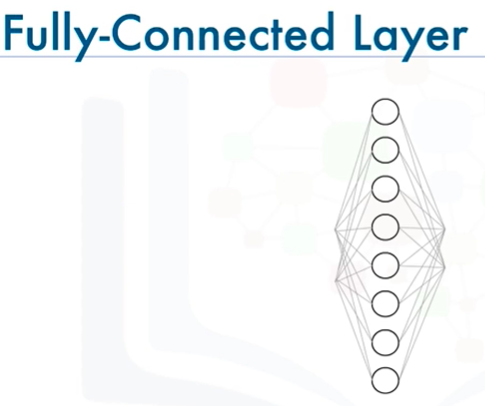
****

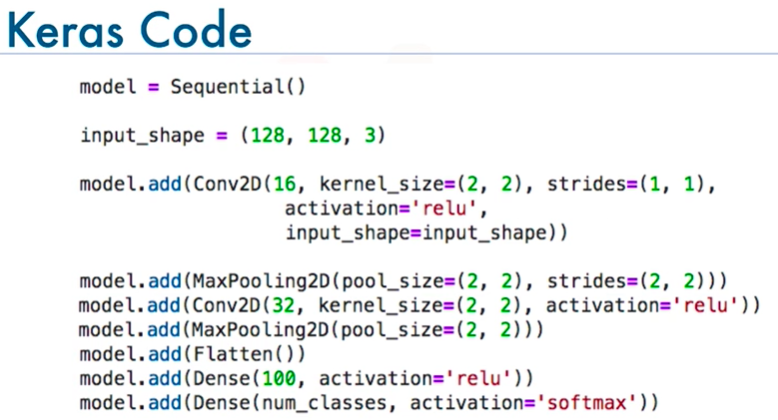
****

* But one question you must be asking yourself at this point is, why would we need to use convolution? Why not flatten the input image into an (n x m) x 1 vector and use that as our input? Well, if we do that, we will end up with a massive number of parameters that will need to be optimized, and it will be super computationally expensive. Also, decreasing the number of parameters would definitely help in preventing the model from overfitting the training data. It is worth mentioning that a convolutional layer also consists of ReLU's which filter the output of the convolutional step passing only positive values and turning any negative values to 0.









**Recurrent Neural Networks**

****

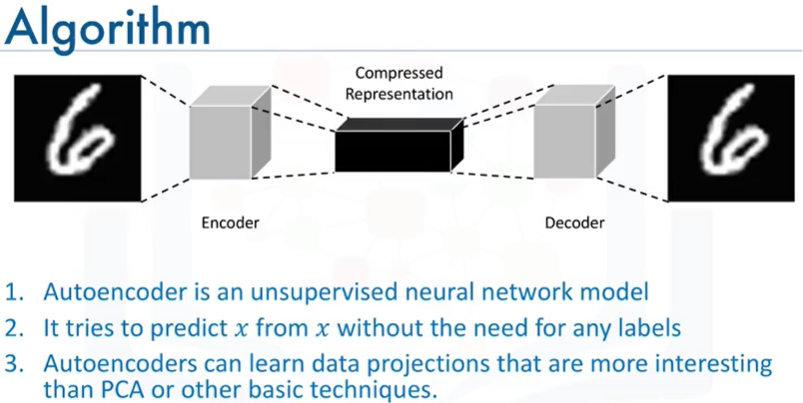
**Recurrent Neural Networks are networks with loops, that don't just take a new input at a time, but also take as input the output from the data point at the previous instance.**

**recurrent neural networks (RNNs)**, which are a type of supervised deep learning model. Here are the key points:

* **RNNs** are designed to handle sequences of data, making them suitable for applications like analyzing scenes in movies, where data points are not independent.
* Unlike traditional neural networks, RNNs have loops that allow them to take the output from the previous time step as input for the current time step.
* This architecture enables RNNs to model patterns and sequences effectively, making them useful for tasks such as text analysis, handwriting recognition, and stock market predictions.
* A popular variant of RNNs is the **long short-term memory (LSTM)** model, which has been successfully applied in various fields, including image generation and video description.

This concludes the overview of recurrent neural networks, and the next topic will cover unsupervised deep learning models, specifically autoencoders.

**Autoencoders**

****

**autoencoders**, an unsupervised deep learning model. Here are the key points:

* **Definition**: Autoencoders are data compression algorithms that learn to compress and decompress data automatically using neural networks.
* **Data Specificity**: They are trained on specific data types, meaning an autoencoder trained on images of cars may not effectively compress images of buildings.
* **Applications**: Common uses include:
  + **Data Denoising**: Removing noise from data.
  + **Dimensionality Reduction**: Simplifying data for visualization.
* **Architecture**: An autoencoder consists of an encoder that compresses the input (e.g., an image) and a decoder that reconstructs the original input.
* **Learning Process**: It uses backpropagation with the target variable set to the same as the input, aiming to approximate an identity function.
* **Comparison with PCA**: Autoencoders can learn more complex data projections than traditional methods like Principal Component Analysis (PCA).
* **Restricted Boltzmann Machines (RBMs)**: A popular type of autoencoder, useful for:
  + Balancing imbalanced datasets.
  + Estimating missing values.
  + Automatic feature extraction from unstructured data.