BREAK THROUGH TECH

# Machine Learning Foundations

Lab 7

Week of July 14

BREAK THROUGH TECH

# Icebreaker: Emoji Code

# Decode the Emojis

**Objective:**

- Decode the messages made entirely of emojis.

**Rules:**

- Each emoji “sentence” represents a phrase or activity.
- You will have 10 seconds to guess each message.
- Type your guesses into the chat.
- Feel free to guess as many times as you like.

# Why Are We Playing Emoji Code?

**It’s harder than you think!**

- Like decoding emojis, computer vision systems interpret visual data to understand and make sense of images and videos.
- Today’s game highlights how we, and machines, use patterns and context to interpret information.
- This skill is fundamental in areas like image recognition, facial recognition, and automated driving systems.

# Decode the Emojis

# Decode the Emojis



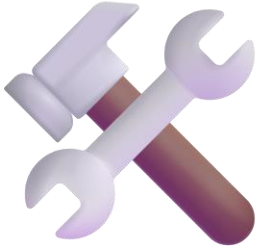


# Decode the Emojis

# Decode the Emojis

# Decode the Emojis

# Decode the Emojis



# Wrap Up

- While it might seem like a simple game to us, interpreting images can be quite complex. For computers, this process of decoding visual information is even more challenging.
- In computer vision, machines must learn to recognize and interpret not just simple icons like emojis but complex real-world visuals—everything from facial expressions in photos to obstacles in video from autonomous vehicles.
- This game was a fun way to illustrate that even for systems designed to ‘see’ and ‘understand,’ the task is not straightforward. Achieving accurate interpretation requires advanced algorithms and significant processing power, highlighting the impressive capabilities and ongoing challenges in the field of AI.

BREAK
THROUGH
TECH

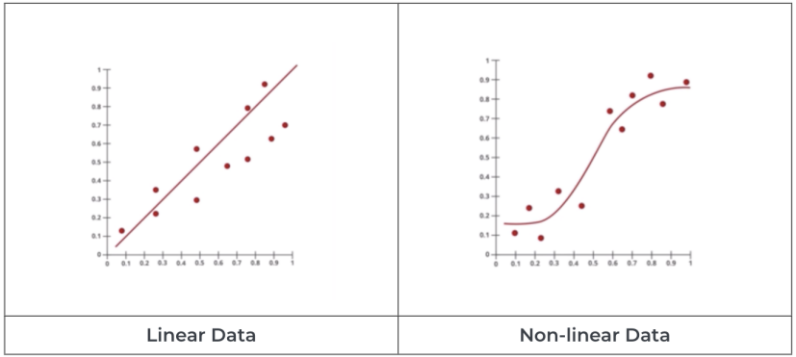
# Week 7 Concept Overview + Q&A

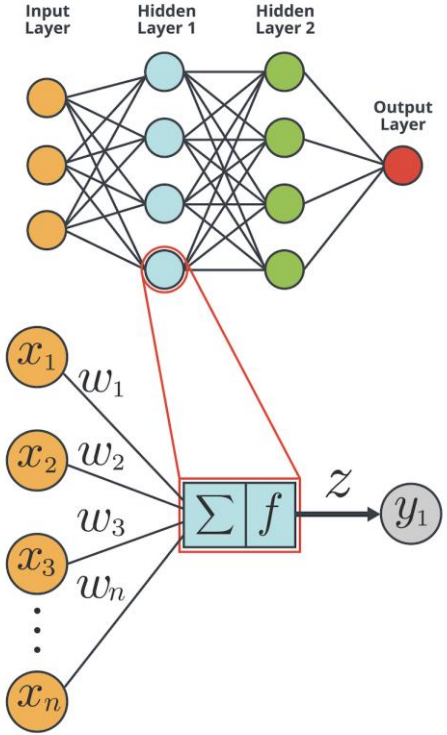
# Week 7 Concept Overview

This week covered a number of topics. To refresh your memory, here is what you've completed:

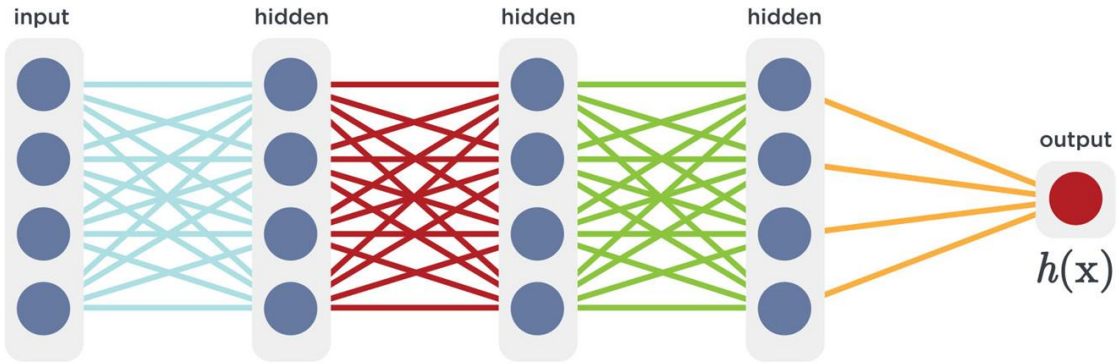
- Explore the design of a neural network using the basic components of network architecture
- See how a neural network is trained and optimized
- Use Keras to implement a neural network to make predictions
- Explore the field of Computer Vision
- Implement a neural network for image classification

# Linear and Nonlinear Transformations



# Neural Networks

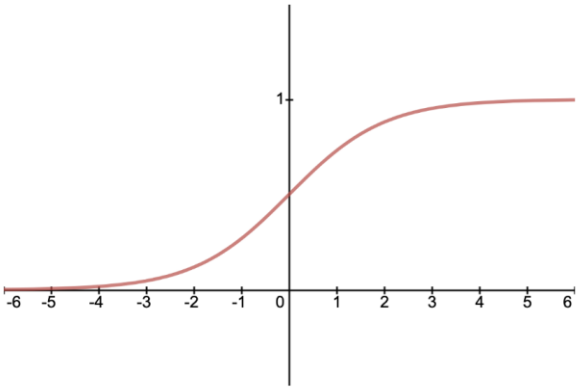



$$h(\mathbf{x}) = \mathbf{W}_4\sigma(\mathbf{W}_3\sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x})))$$

# Activation Functions

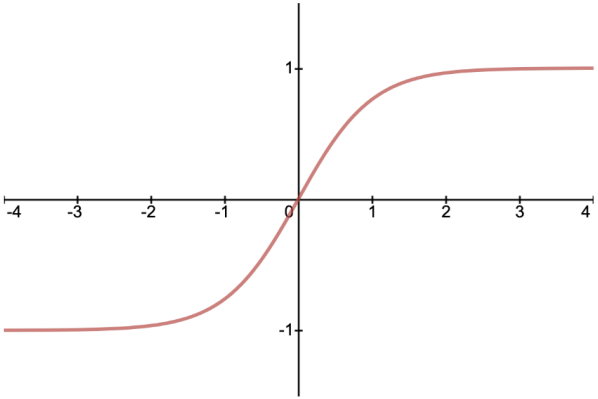
**Sigmoid function**

Sigmoidal unit: $\sigma(z) = \frac{1}{1+e^{-z}}$



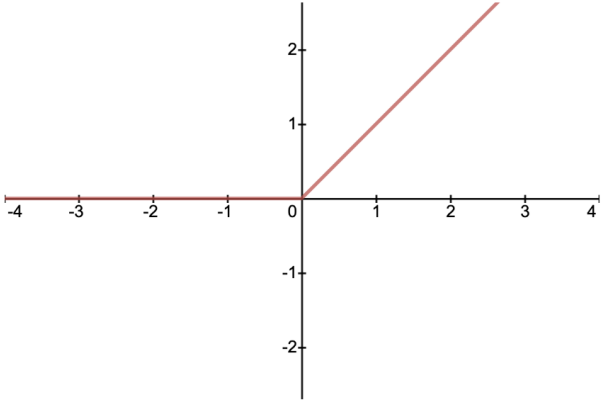

**Hyperbolic tangent/tanh function**

tanh: $\sigma(z) = \tanh(z)$




**ReLU function**

ReLU: $\sigma(z) = \max(z, 0)$



# Loss Functions

- **Regression**

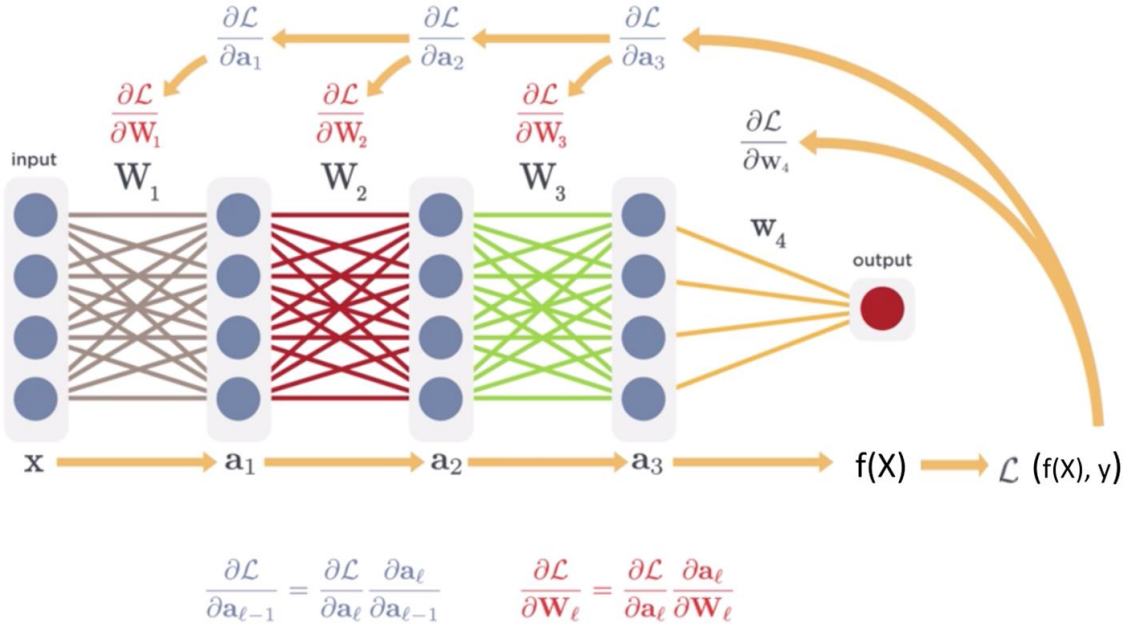
$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{h}(\mathbf{x}_i), y_i) = \frac{1}{n}\sum_{i=1}^{n} \left(\mathbf{h}(\mathbf{x}_i) - y_i\right)^2$$

- **Classification**

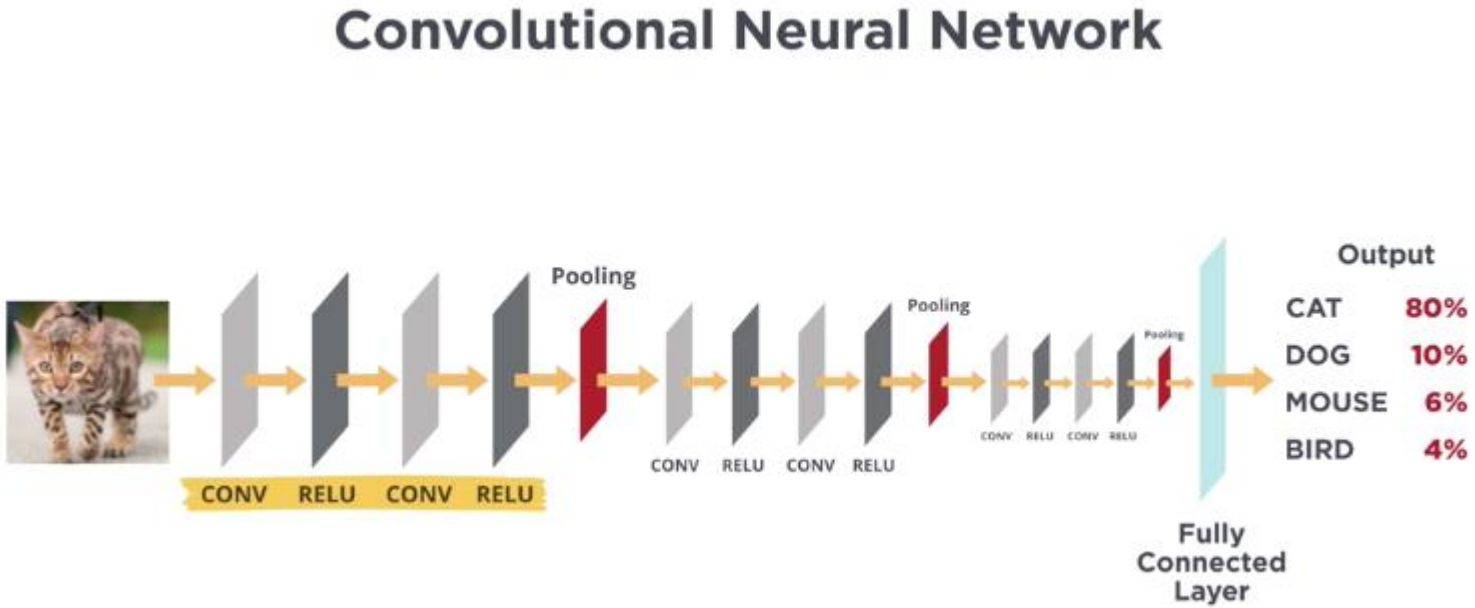
$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} \log\left(\frac{\exp\left([\mathbf{h}(\mathbf{x}_i)]_{y_i}\right)}{\sum_{j=1}^{k}\exp\left([\mathbf{h}(\mathbf{x}_i)]_j\right)}\right)$$

# Neural Network Training and Optimization

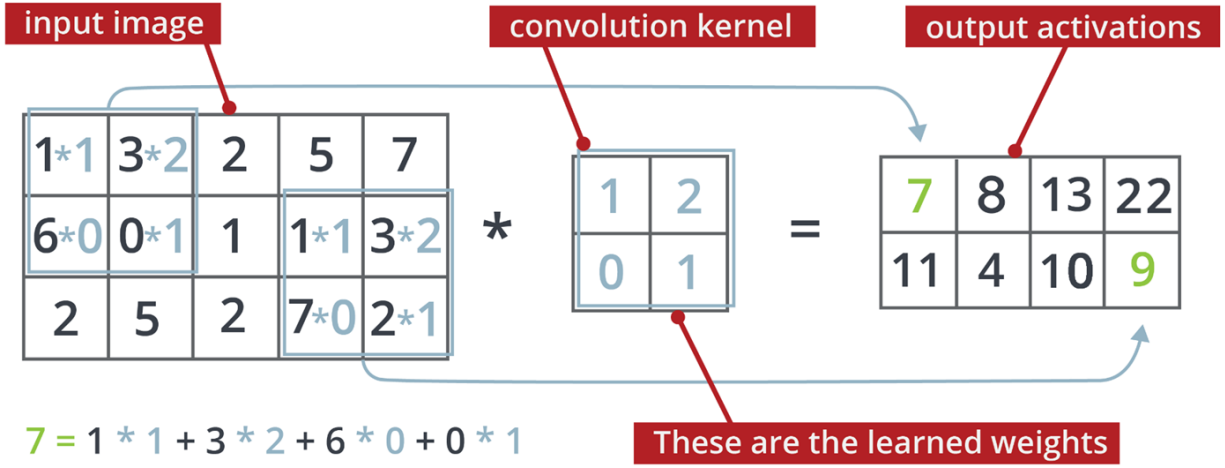
Forward and backward propagation



# Computer Vision:
# Neural Network Architecture for Image Data



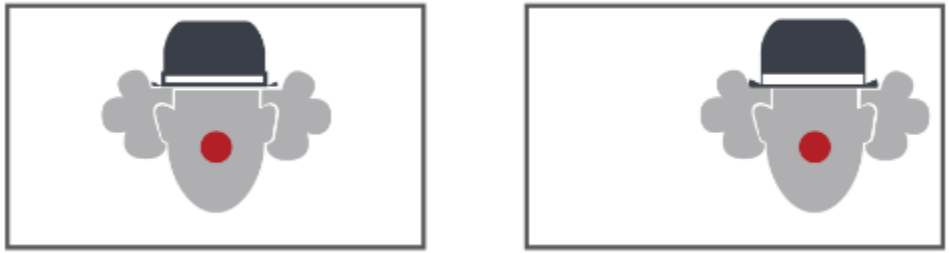
# Convolution Operation



# Pooling Layer

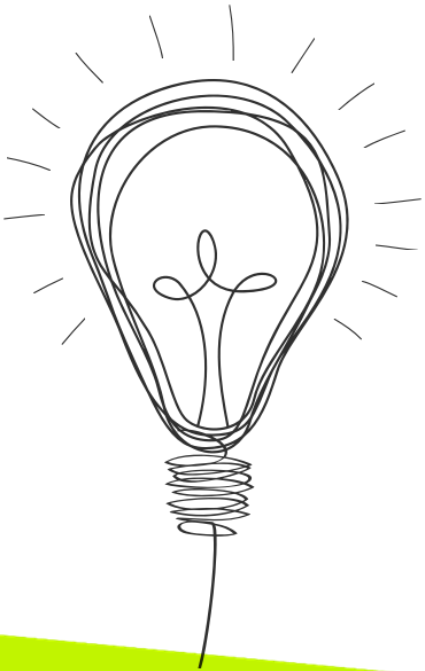
Introduces translation invariance:

- Translation invariance is the ability to withstand small shifts of an object in a photo.
- Pooling layers reduce the dimensions of the feature maps produced by a convolutional layer. They downsample the feature maps by extracting the most important features from different spatial locations, thereby introducing translation invariance.
- A good model that uses pooling layers would be able to recognize that the image is the same even when in a different position (i.e., a clown centered in a photo and the same clown on the right in another photo).



# Questions & Answers

What questions do you have about the online content this week?



BREAK THROUGH TECH

# Breakout Groups: Big Picture Questions

# Big Picture Questions

You have 15  minutes to discuss the following questions within your breakout groups:

- Explain Deep learning.
- What is a neural network and why is it needed to solve some ML problems?
- What are the key differences between neural networks and other machine learning models?
- What is backpropagation, and why is it crucial for training a neural network?
- What are some limitations/common challenges faced when training neural network for computer vision tasks?
- Describe the core components of a CNN and how it differs from a traditional NN.
- How does the process of image classification work in a CNN?

# Big Picture Responses - Explain Deep learning

Deep learning is a subfield of machine learning that deals with algorithms inspired by the structure and function of the brain's neural networks. It is called "deep" learning because it involves neural networks with many layers (deep architectures) that allow the model to learn hierarchical representations of data.

Deep Architectures: Unlike traditional machine learning models that may only have a few layers, deep learning models have many layers (hence the term "deep"). These layers enable the model to automatically learn features at different levels of abstraction, starting from simple features at the lower layers to more complex features at the higher layers.

Learning from Data: Deep learning models are trained using large amounts of labeled data. During training, the model adjusts its weights (parameters) based on the input data and the expected output, aiming to minimize the difference between its predictions and the actual labels.
This process is typically done using optimization algorithms like stochastic gradient descent.

# Big Picture Responses - What is a neural network and why is it needed to solve some ML problems?

A neural network is a computational model inspired by the structure and functioning of biological neural networks, such as the human brain. It consists of interconnected nodes called neurons organized in layers.
Each neuron receives input, processes it using a mathematical function, and passes the output to the next layer of neurons.
Each neuron typically applies a non-linear activation function to introduce non-linearity into the model, enabling it to learn complex patterns.
They are essential in machine learning for their ability to learn complex patterns from data, handle non-linear relationships, and achieve high performance on challenging tasks where traditional methods may struggle.
Their flexibility and scalability make them a powerful tool for solving a wide range of real-world problems.
Neural networks are particularly effective for solving complex problems in machine learning for several reasons:

1. Non-Linearity: Many real-world problems, such as image and speech recognition, involve non-linear relationships between inputs and outputs. Neural networks with non-linear activation functions can model these relationships more effectively compared to linear models.
2. Representation Learning: Neural networks can automatically learn and extract intricate features from raw data, reducing the need for manual feature engineering. This ability is crucial for tasks where the underlying patterns are complex and not easily represented by handcrafted features.
3. Scalability: Neural networks can scale with the amount of data and computational resources available. By increasing the number of neurons, layers, and training data, neural networks can potentially achieve higher accuracy and better performance on challenging tasks.
4. Flexibility: Different types of neural network architectures (e.g., convolutional neural networks for images, recurrent neural networks for sequential data) can be tailored to specific types of data and tasks, making them versatile across a wide range of applications.

# Big Picture Responses - differences between neural networks vs other machine learning models?

Neural networks differ from traditional machine learning models due to their architecture, learning approach, and ability to handle complex data relationships. Here are some key differences:

**1. Architecture**

- Neural Networks (NNs):
  - Structure: Organized into layers of interconnected nodes (neurons).
  - Depth: Can be deep, consisting of multiple hidden layers (hence deep learning).
  - Connections: Neurons within layers are fully connected to neurons in subsequent layers (in fully connected networks), or sparsely connected in architectures like convolutional or recurrent neural networks.

VS

- Traditional Machine Learning Models:
  - Structure: Typically simpler, consisting of a fixed number of input features and parameters.
  - Depth: Shallow, with few layers (if any), often consisting of a single layer (as in linear models) or a few layers (as in decision trees).

**2. Learning Approach**

- Neural Networks (NNs):
  - Learning: Learn through optimization algorithms (e.g., gradient descent) that adjust weights and biases iteratively to minimize error.
  - Feature Learning: Automatically learn hierarchical representations from raw data, reducing the need for manual feature engineering.

VS

- Traditional Machine Learning Models:
  - Learning: Typically involve learning a function that maps input features directly to outputs.
  - Feature Engineering: Often rely on manual feature selection and engineering to extract relevant information from raw data.

**3. Handling of Data**

- Neural Networks (NNs):
  - Data Complexity: Well-suited for complex data such as images, audio, and text due to their ability to learn intricate patterns and representations.
  - Non-Linearity: Can model non-linear relationships between inputs and outputs effectively.

VS

- Traditional Machine Learning Models:
  - Data Types: Effective for structured data with clear feature representations.
  - Linear Relationships: Often assume or approximate linear relationships between variables.

## Big Picture Responses - What is backpropagation, and why is it crucial for training a neural network?

Backpropagation is crucial for training neural networks because it enables them to learn from data by adjusting their internal parameters (weights and biases) based on the error in their predictions. This iterative process of forward pass, error computation, gradient calculation, and weight update allows neural networks to improve their performance on tasks ranging from image and speech recognition to natural language processing and beyond.

1. **Learning:** Backpropagation allows the neural network to learn from examples by adjusting its weights and biases based on the error it makes during training. By propagating the error backward through the network, it identifies which weights contributed most to the error and adjusts them accordingly.

2. **Efficiency:** Neural networks with multiple layers (deep networks) are capable of learning complex patterns and representations from data. Backpropagation efficiently computes the gradients needed to update the weights across all layers, enabling deep networks to learn hierarchical representations.

3. **Flexibility**: Backpropagation is not tied to specific architectures and can be applied to various types of neural networks, including convolutional neural networks (CNNs) for image data and recurrent neural networks (RNNs) for sequential data.

4. **Scalability:** With the advent of frameworks like TensorFlow and PyTorch, implementing backpropagation is straightforward, allowing researchers and engineers to experiment with deep neural network architectures and train models on large-scale datasets.

5. **Improvement:** Over time, advancements in optimization algorithms (e.g., SGD, Adam, RMSprop) and regularization techniques (e.g., dropout, batch normalization) have improved the efficiency and convergence rates of backpropagation, making it even more effective for training neural networks.

# Big Picture Responses-What are some limitations when training neural network for computer vision tasks?

Despite its success, deep learning also presents challenges, such as the need for large amounts of labeled data, computational resources for training large models, and interpretability of the learned representations.

1. **Large Amounts of Data:** Neural networks for computer vision often require extensive datasets for training, especially deep architectures like convolutional neural networks (CNNs). Acquiring and annotating large-scale datasets can be time-consuming and costly.
2. **Computational Resources:** Training deep neural networks on large datasets requires significant computational power and memory. This includes GPUs or TPUs for faster processing, which may not be readily available to all researchers or practitioners.
3. **Overfitting:** Neural networks are prone to overfitting, where the model performs well on training data but poorly on unseen data. Techniques like dropout, data augmentation, and regularization are used to mitigate overfitting, but balancing model complexity and generalization remains a challenge.
4. **Architecture Design:** Choosing an appropriate neural network architecture for a specific computer vision task can be non-trivial. There are various architectures (e.g., CNNs, RNNs, Transformers) each suited to different types of visual data and tasks. Designing an architecture that balances complexity and performance requires domain expertise and experimentation.
5. **Hyperparameter Tuning:** Neural networks have several hyperparameters (e.g., learning rate, batch size, number of layers) that significantly impact training performance. Finding the optimal set of hyperparameters through grid search, random search, or automated techniques like Bayesian optimization can be time-consuming and computationally expensive.
6. **Gradient Vanishing/Exploding:** In deep neural networks, especially in recurrent networks or very deep CNNs, gradients can diminish (vanish) or explode during backpropagation, affecting training stability and convergence. Techniques like careful weight initialization, gradient clipping, and using appropriate activation functions (e.g., ReLU) help mitigate these issues.
7. **Interpretability:** Deep neural networks are often considered black boxes, making it challenging to interpret their decisions or understand how they arrived at a particular prediction. Techniques like visualization of activations, gradient-based methods for saliency maps, and model distillation are used to improve interpretability.
8. **Domain Shift:** Neural networks trained on one dataset may not generalize well to different distributions of data (domain shift). Transfer learning and domain adaptation techniques help mitigate this issue by leveraging knowledge learned from related tasks or domains.
9. **Preprocessing and Augmentation:** Proper preprocessing of input data (e.g., normalization, resizing) and augmentation (e.g., rotation, flipping) are crucial for training robust models. However, finding the right balance of preprocessing steps and augmentation techniques can impact model performance significantly.
10. **Label Noise and Quality:** The quality and accuracy of labels in the training dataset can affect the performance of neural networks. Noisy or incorrect labels can lead to biases in the model, impacting its ability to generalize to unseen data.

## Big Picture Responses – Describe the core components of a CNN and how it differs from a traditional NN.

1. Structure and Connectivity:
   - CNNs: CNNs are structured to preserve the spatial relationships in the input data (e.g., pixels in images). They use local connectivity (via convolutional filters) and parameter sharing to efficiently learn spatial hierarchies of features.
   - Traditional NNs: Traditional neural networks are fully connected, where each neuron in one layer is connected to every neuron in the subsequent layer, without considering the spatial arrangement of inputs.
2. Feature Learning:
   - CNNs: CNNs excel at learning hierarchical representations of spatial data. They automatically learn features at different levels of abstraction (e.g., edges, textures, objects) from raw input data.
   - Traditional NNs: Traditional neural networks rely on manually engineered features or feature extraction methods before being fed into the network.
3. Applications:
   - CNNs: CNNs are primarily used for tasks involving grid-like data such as images and videos. They have achieved state-of-the-art results in image classification, object detection, and segmentation.
   - Traditional NNs: Traditional neural networks are versatile and can be applied to various types of data and tasks, but they may not perform as well on tasks that require spatial relationships and hierarchical feature learning.
4. Parameter Efficiency:
   - CNNs: Due to parameter sharing in convolutional layers, CNNs are more parameter efficient and require fewer parameters compared to fully connected networks of similar depth.
   - Traditional NNs: Traditional neural networks have more parameters per layer, leading to increased computational and memory requirements, especially for deep architectures.

In summary, CNNs are specialized neural networks designed for handling spatial data efficiently. Their architecture, which includes convolutional and pooling layers, enables them to learn hierarchical features directly from raw input, making them highly effective for computer vision tasks where understanding spatial relationships is crucial. This distinguishes them from traditional neural networks, which are more general-purpose and may require manual feature engineering for complex data types like images.

- **Big Picture Responses -** How does the process of image classification work in a CNN?

Image classification in a Convolutional Neural Network (CNN) involves the network predicting a class label for an input image. Here's how the process typically works in a CNN:

1. Input Image:
   - Input: An image is fed into the CNN. Each image is typically represented as a 3D tensor (height × width × channels), where channels usually represent RGB color channels (red, green, blue).
2. Convolutional Layers:
   - Feature Extraction: The CNN begins with a series of convolutional layers. Each convolutional layer applies a set of learnable filters (kernels) to the input image.
   - Local Receptive Field: Each filter detects specific features (such as edges or textures) within its local receptive field of the input image.
   - Feature Maps: Multiple filters in each convolutional layer produce multiple feature maps, each highlighting different aspects of the input image.
3. Activation Function:
   - Non-linearity: After each convolution operation, an activation function (typically ReLU) is applied element-wise to introduce non-linearities into the model, allowing it to learn complex relationships in the data.
4. Pooling Layers:
   - Downsampling: Following some convolutional layers, pooling layers (e.g., max pooling) reduce the spatial dimensions of the feature maps while retaining the most important information.
   - Translation Invariance: Pooling helps achieve translation invariance by aggregating local features, making the CNN robust to variations in object position within the image.
5. Flattening:
   - Vectorization: Once the spatial hierarchy of features has been captured through convolution and pooling, the resulting feature maps are flattened into a 1D vector. This flattening process prepares the data to be fed into fully connected layers.
6. Fully Connected Layers:
   - Classification: The flattened features are passed through one or more fully connected layers. These layers integrate the features and learn to associate them with specific classes.
   - Output Layer: The final layer of the CNN is typically a softmax layer, which outputs probabilities for each class. The class with the highest probability is considered the predicted class label for the input image.
7. Loss Function and Optimization:
   - Loss Calculation: The predicted probabilities are compared with the actual labels (ground truth) using a loss function (e.g., categorical cross-entropy).
   - Gradient Descent: Backpropagation and gradient descent are used to optimize the network's weights and biases. This process adjusts the parameters to minimize the loss, improving the network's ability to classify images accurately.

BREAK THROUGH TECH

# Class Discussion

BREAK
THROUGH
TECH

# Break

BREAK THROUGH TECH

# Breakout Groups: Lab Assignment

# Lab 7

In this lab, you will:

- Define your ML problem
- Import image data and split data into training and test data sets
- Inspect and visualize the data
- Prepare your data so that it is ready for modeling
- Construct and train a convolutional neural network model
- Evaluate the model's performance on the training and test data

# Lab 7

## Lab 7: Implementing a Convolutional Neural Network Using Keras

In [ ]:
```
import tensorflow.keras as keras
import math
import time
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

A very common problem in computer vision is recognizing hand-written digits. The images of numerals are commonly used by data scientists and machine learning experts to train supervised learning models that specialize in decoding human handwriting. This is a classic problem that is often used in exercises and documentation. In this lab, you will train a convolutional neural network to classify hand-written digits. You will complete the following tasks:

1. Define your ML problem:
    - Define the label - what are you predicting?
    - Identify the features
2. Import the data and split the data into training and test data sets
3. Inspect and visualize the data
4. Prepare your data so that it is ready for modeling.
5. Construct a convolutional neural network
6. Train the convolutional neural network.
7. Evaluate the neural network model's performance on the training and test data.

For this lab, use the demo *Implementing a Neural Network in Keras* that is contained in this unit as a reference.

BREAK THROUGH TECH

# Working Session 1 Debrief

# Lab Debrief

So far,

- What did you enjoy about this lab?
- What did you find difficult about this lab?
- What questions do you still have about this lab?

BREAK THROUGH TECH

# Working Session 2 Debrief

# Lab Debrief

- What did you enjoy about this lab?
- What did you find hard about this lab?
- What questions do you still have about this lab?
- How did you approach problem-solving during the exercise?
- What would you do differently if you were to repeat the exercise?

BREAK
THROUGH
TECH

# Concluding Remarks

# Concluding Remarks

- Key takeaways
- Additional resources

# Next week

In the following week, you will:

- Explore the NLP pipeline
- Use various NLP preprocessing techniques to convert text to data suitable for machine learning
- Understand how vectorizers are used to convert text into numerical features
- Explore how word embeddings are used to convert text into numerical features without losing the underlying semantic meaning
- Discover how deep neural networks are used in the NLP field
- Implement a feedforward neural network for sentiment analysis
- Create and implement a project plan to solve a machine learning problem

And in the lab, you will:

- Build your DataFrame
- Define your ML Problem
- Perform exploratory data analysis to understand your data
- Define your project plan to implement an ML model for your problem
- Implement your project plan
- Create a portfolio to showcase your project

BREAK THROUGH TECH

# Content + Lab Feedback Survey

# Content + Lab Feedback Survey

To complete your lab, please answer the following questions about BOTH your online modules and your lab experience. Your input will help pay it forward to the Break Through Tech student community by enabling us to continuously improve the learning experience that we provide to our community.

Thank you for your thoughtful feedback!

https://forms.gle/eUQQZgS6BPRpqgZ7A