Check-In Code:

cnns

Intro to CV: Convolutional Neural Networks (CNNs)

Presented by Jackie, Milan, and Weiji

ACM AI

Please Check In!

Checking in helps you get membership points & helps us track attendance

Intro to CV: Convolutional Neural Ne...

Check-in code: cnns

today's agenda

1 **Review**
Computer Vision & MNIST

2 **Why is MLP not Optimal?**
Overfitting, Translational Invariant, Spatial Information

3 **Introduction to CNNs**
Components, Layers, Optimization

4 **Examples of CNNs**
The 5 architectures to know

ACM AI

Interactive Notebook

https://acmurl.com/cv2-interactive

ACM AI

Review

General ML with MNIST

ACM AI

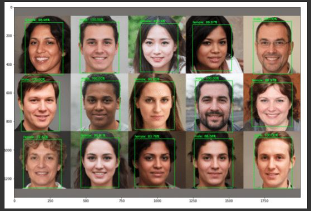
What is Computer Vision?

Definition and Application

- Computer Vision is a field of AI that enables computers and systems to derive meaningful information from digital images, vidoes, and other visual inputs - and take actions or make recommendations based on that information.
- Autonomous Vehicles, Facial Recognition, Medical Imaging

Computer Vision Tools

- PyTorch - popular ML framework for AI applications



ACM AI

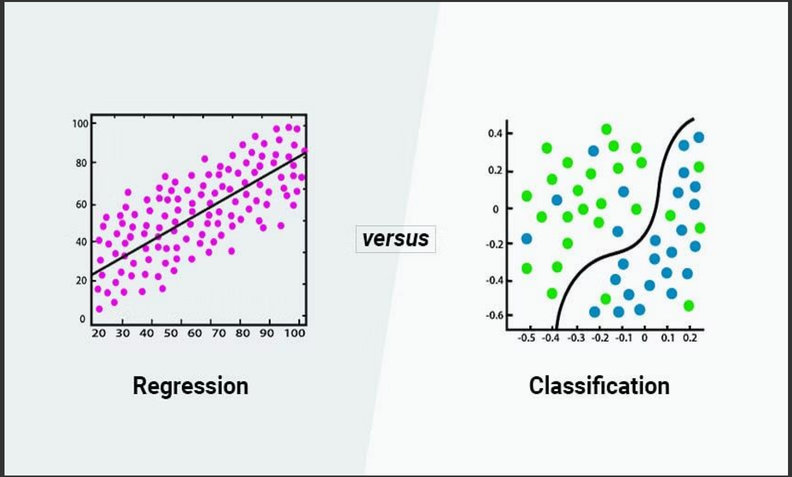
General ML Problems

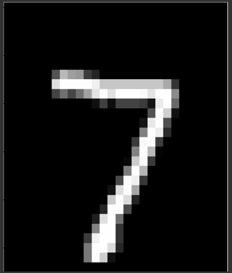
Formulating a (supervised) ML problem

ACM AI

General Machine Learning Problems

- In general, ML can be applied to many different situations, and their formulations are broad and varied - from decision-making (chess) to holding conversations (chatbots)
- At the core, it's about designing a function mapping input to output
- Two major types: Regression and Classification, difference is regression outputs a continuous variable while classification outputs a discrete variable





28x
28

ML
Model

$\xrightarrow{\hspace{4em}}$ 7

10
possibl
e labels

ACM AI

Neural Networks

ACM AI

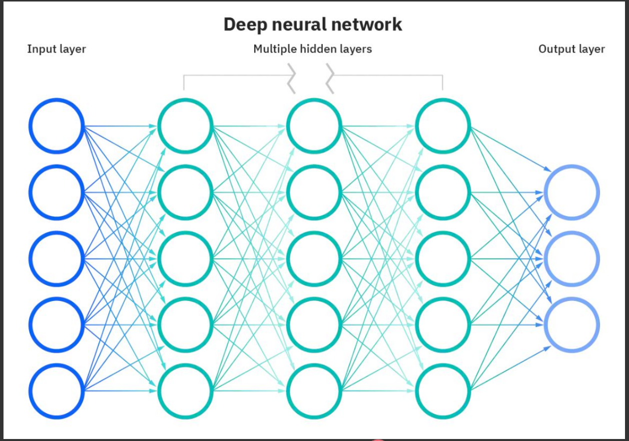
Neural Networks: Definition

Definition

- Mimics the brain through a set of algorithms
- A neural network is comprised of 4 main components: inputs, weights, a bias, and output
- Expressive nonlinear function approximators
  - Given an arbitrary number of “neurons”, neural networks can approximate any function
  - Introduce nonlinearities through activation functions

$$\sum_{i=1}^{m} w_i x_i + bias = w_1 x_1 + w_2 x_2 + w_3 x_3 + bias$$

- Deep Learning: a subset of ML techniques based on multiple layers of neural networks



ACM AI

Neural Networks: Components

Training

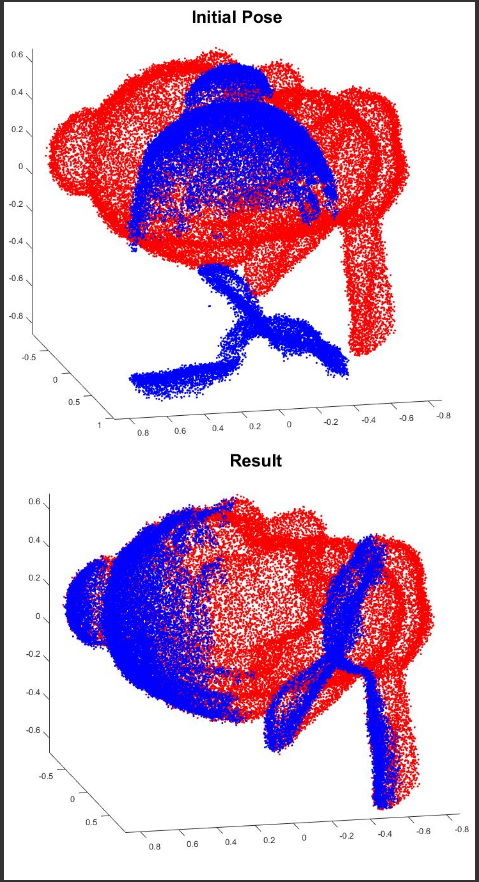
- Composed of 5 essential steps: defining architecture, forward propagation, calculating loss, **backward propagation**, and updating weights using learning rate
- Gradients and Autograd are essential to backward propagation

Gradients

- Derivatives of the loss function along every dimension
- Used to update the weight using a learning rate to reduce the loss and train the neural network.

Loss Functions

- We need a method of evaluating how correct/wrong a given answer is
- We apply a softmax function to predict probabilities
- A method of evaluating how well specific algorithms models the given data
    - Large loss function = predictions deviate too far from the results
    - Using optimization functions, loss functions learns to reduce the error in prediction

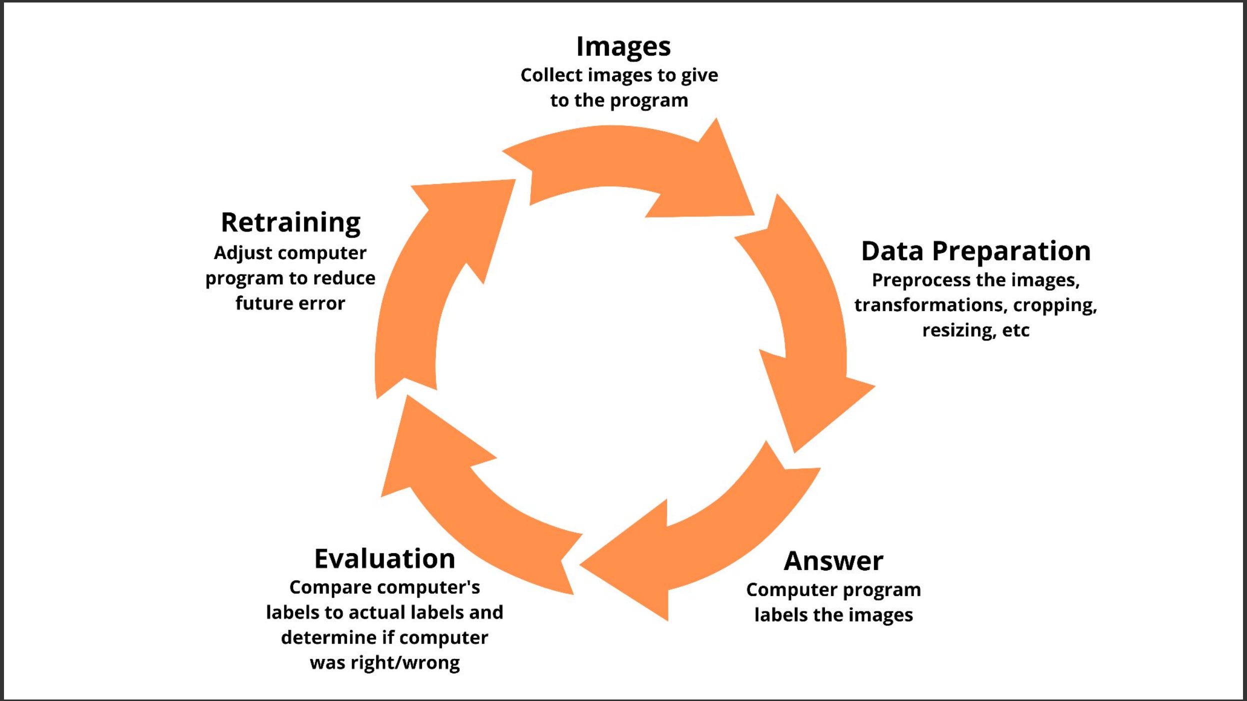


ACM AI

How can we get computers to identify objects in images?

ACM AI

The Idea

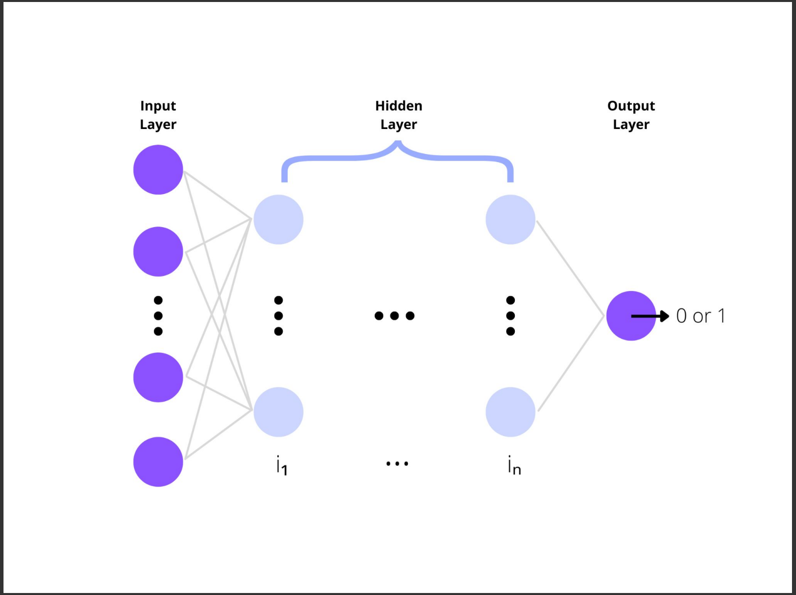

Images
Collect images to give to the program
Data Preparation
Preprocess the images, transformations, cropping, resizing, etc
Answer
Computer program labels the images
Evaluation
Compare computer's labels to actual labels and determine if computer was right/wrong
Retraining
Adjust computer program to reduce future error

ACM AI

Multilayer Perceptron: Definition

Definition

- A fully connected class of feedforward neural networks
- Neural networks with at least three layers
- These three layers are input layer, hidden layer, and output layer
- It uses one perceptron for each input (with images, each perceptron is a pixel)
- In an MLP, the layers are fully connected to each other and each node is connected to each other, with their individual weights and bias.



$$a = \phi\left(\sum_j w_j x_j + b\right)$$

ACM AI

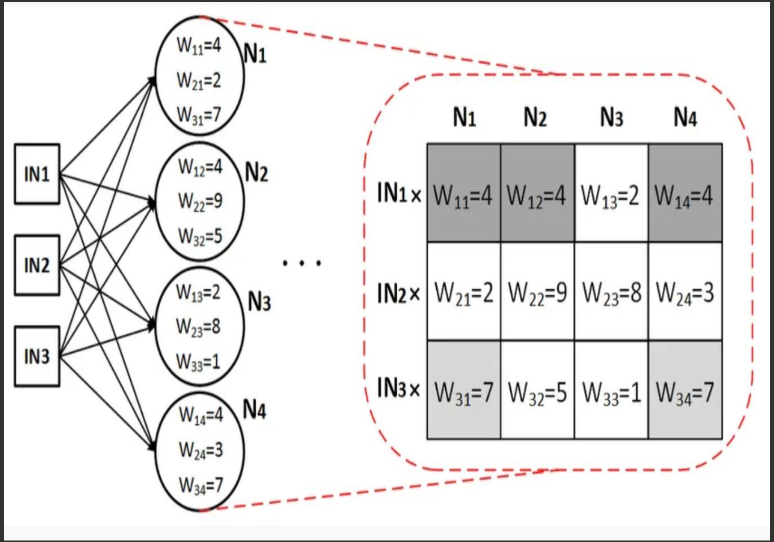
Why is MLP not optimal?

ACM AI

Overfitting

Problem

- Because the MLP is so densely connected, with each node connected to each other and each layer fully connected to each other, the amount of weights become unmanageable for large images - there's a lot of redundancy and inefficiency.
- Overfitting occurs which means it loses the ability to generalize
- MLP is also not translational invariant - it reacts differently to the shifted version of an image
- Spatial information is also lost when the image is flattened (matrix to vector) into an MLP because MLP takes in vector as an input
- Thus, while MLP can be used for image classification, it becomes very difficult to use when there's more complex images.
- So what's the solution?



	N1	N2	N3	N4
IN1 ×	$W_{11}=4$	$W_{12}=4$	$W_{13}=2$	$W_{14}=4$
IN2 ×	$W_{21}=2$	$W_{22}=9$	$W_{23}=8$	$W_{24}=3$
IN3 ×	$W_{31}=7$	$W_{32}=5$	$W_{33}=1$	$W_{34}=7$

ACM AI

Intro to CNN

Architecture, Layers, Optimization

ACM AI

CNN: Why is it better than MLP?

CNN Input

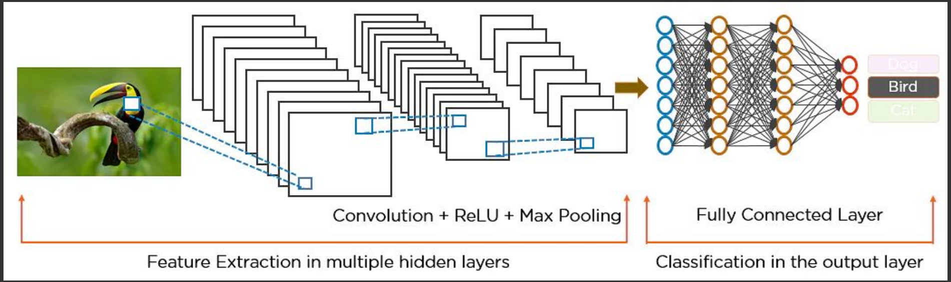
- CNN takes in tensor (matrices) as an input so it's capable of understanding spatial relation (relation between nearby pixels of image)
- CNN can develop an internal representation of a two-dimensional image. This allows the model to learn position and scale in variant structures in the data

Spatially Invariant

- Not sensitive to the position of the object in the picture
- It's able to find patterns regardless of where the pattern is at in the image

Faster computation and less memory usage

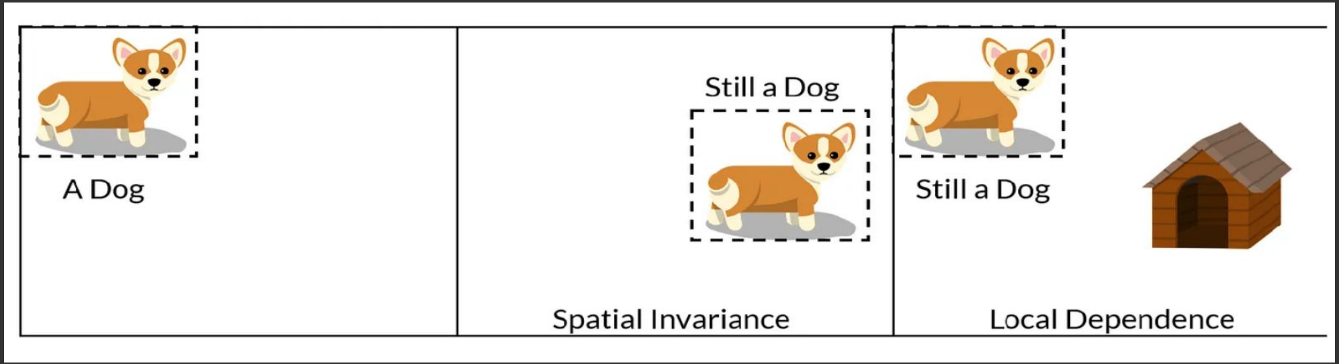
- The weights are smaller and shared - so less wasteful compared to MLP and is more effective
- Layers are also more sparsely connected allowing flexibility and preventing overfitting


Dog
Bird
Cat
Convolution + ReLU + Max Pooling
Fully Connected Layer
Feature Extraction in multiple hidden layers
Classification in the output layer

ACM AI

CNN: How does it work?

- CNN’s leverage the fact that nearby pixels are more strongly related than distant ones. It analyzes the influence of nearby pixels by using a **filter/Kernel** and we can move this across the image from top left to bottom right. For each point on the image, a value is calculated based on the filter using a **convolution operation**
- The filters in CNN allows parameter sharing, weight sharing so that the filter looks for a specific pattern, and it’s location invariant - so it can find the pattern anywhere in an image
- It can also find the similar pattern, even if the object is somewhat rotated/tilted using a concept called **Pooling**, which makes the CNN more robust to changes in the position of the feature - this is called local translation invariance



ACM AI

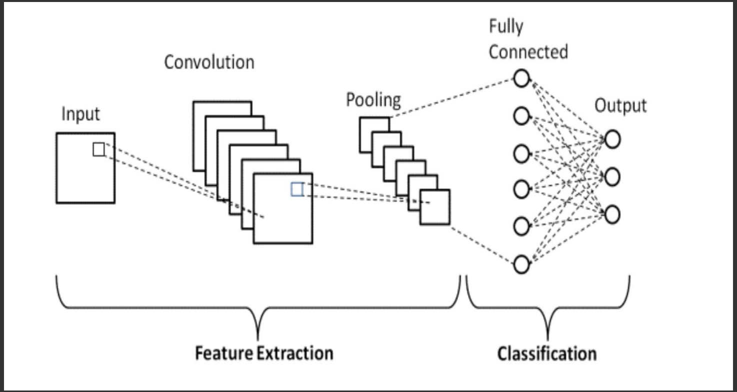
CNN: Architecture

- Convolution in CNN refers to the mathematical function. In math, a convolution is defined as when two functions are multiplied to form a third function that expresses how one function is modified by the other.
- In terms of convolution application to images, this means two images (represented as matrices) can be multiplied together to produce an output that can be used to extract features from the image

- Two main parts to CNN architecture
  - Feature Extraction
  - Classification

- Feature Extraction - process of transforming raw data into numerical features that can be processed while preserving the information in the original data set
  - Done through convolution + pooling layer

- Classification - predicts the class the image belongs in
  - Done through the fully connected layer



- CNN is comprised of three main layers: Convolutional layer, pooling layer, and fully connected layer. Stacked together, this is the architecture

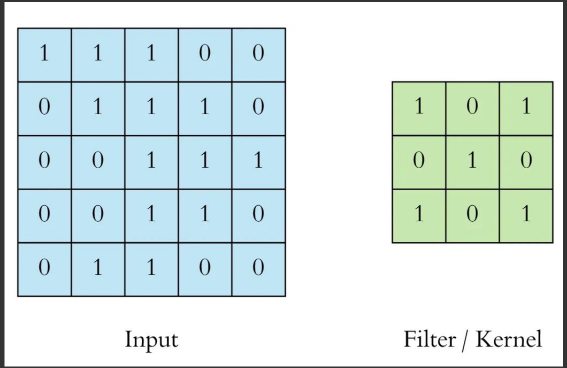
ACM AI

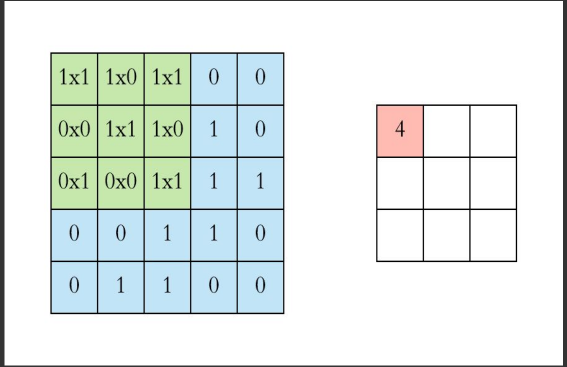
CNN: Convolutional Layer

- Core building block of the CNN, carries the main portion of the computational load
- Convolution is applied on the input data using a convolution filter/kernel to produce a feature map.

Operation

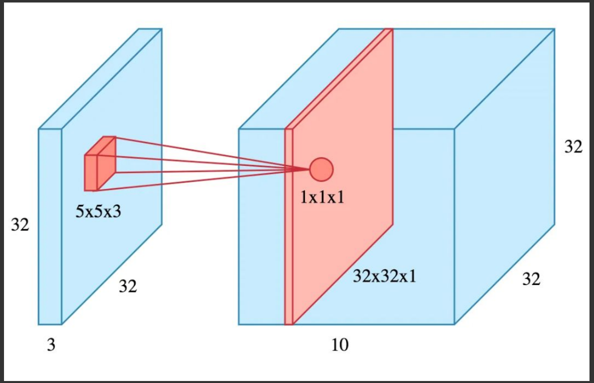
- The convolution operation is performed by sliding the filter over the input. At every location, we do an element-wise matrix multiplication and sum the result. The sum goes into the feature map. The green area where the convolution takes place is called the receptive field.
- This is ultimately aggregated into a feature map
- Multiple convolutions are performed on an input, each using a different filter and resulting in a distinctive feature map. These feature maps are all stacked together and that becomes the final output of the convolution layer

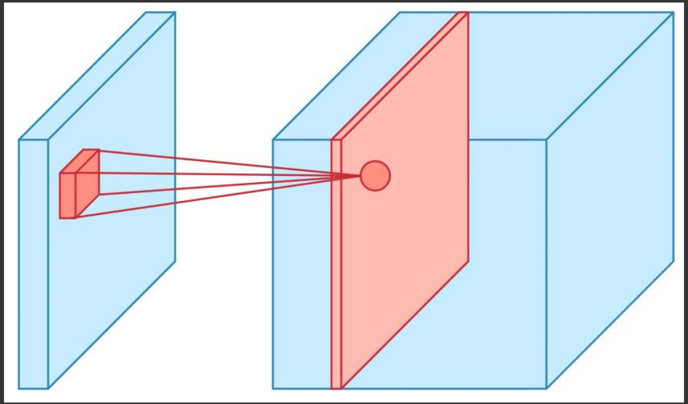




ACM AI

CNN: Visualization of Convolution (single filter)





Example of a 3D representation

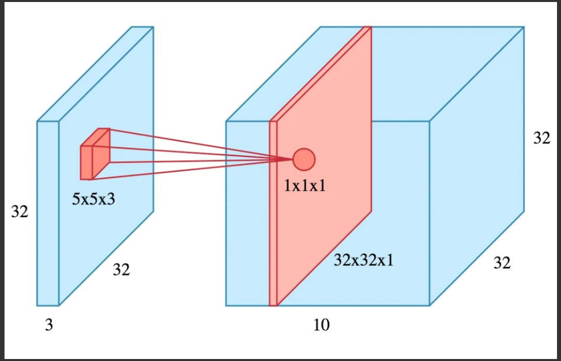
- 32x32xe3 image using a filter of 5x5x3
- The filter is set at a particular location and then it slides over the entire image, performing the convolution, and aggregating the result in a feature map (which is size 32x32x1 as shown in the red slice on the right)

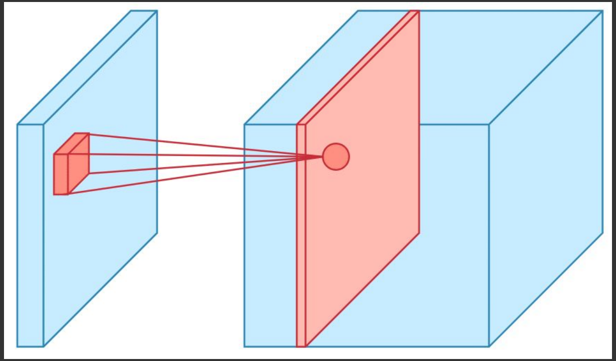
Multiple Filters

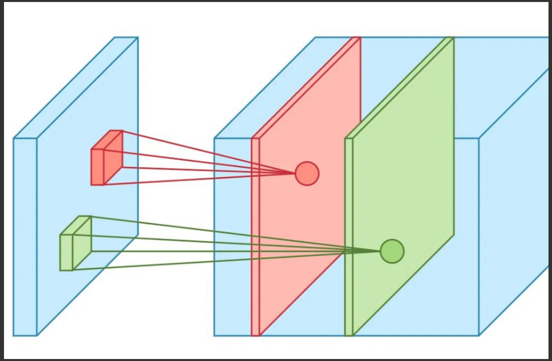
- The picture shows only a single filter, but if we use 10 different filters, we would have 10 feature maps (all of them size 32x32x1) and stacking them along the depth will give us the final output: a convolution layer of 32x32x10

ACM AI

CNN: Visualization of Convolution (Multiple Filters)







Summary

- Input Data is taken
- Filter is applied over input data
- Convolution Operation is performed by filter on input data
- The results are aggregated on a feature map
- Multiple filters are used to aggregate multiple feature maps
- Each feature map is stacked together to output the final result: a convolution layer

ACM AI

CNN: Convolutional Layer Hyperparameters

These are the choices we need to make when creating a convolutional layer.

Filter Size

- Example earlier uses a 3x3 filter, which is typical, but other sizes are also used depending on the application

Filter Count

- A power of two anywhere between 32 and 1024. Using more filters results in a more powerful model, but also risk overfitting due to increased parameter count. Usually, we start with a small number of filters as the initial layers and increase the count as we go deeper into the network

Stride

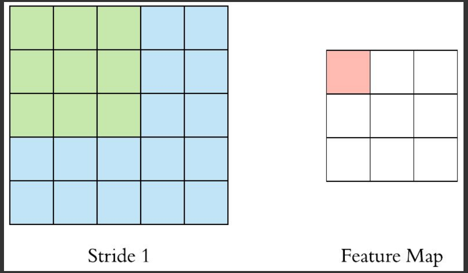
- Specifies how much we move the convolution filter at each step. Default value is 1

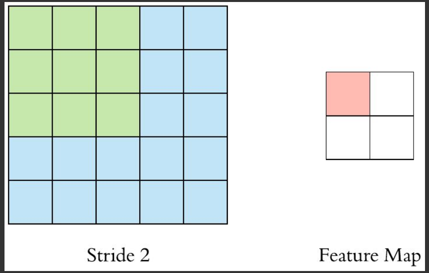
Padding

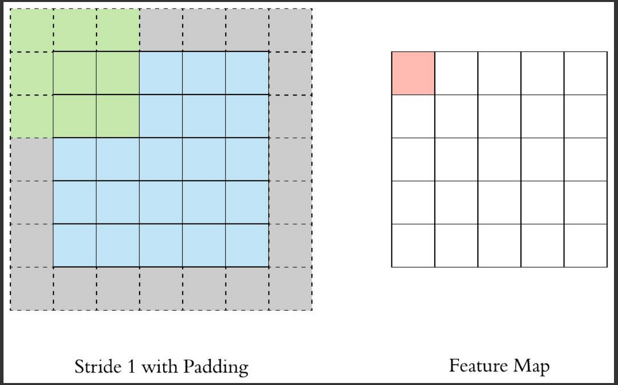
- Refers to surrounding the input with zeroes or the values on the edge to maintain dimensionality (preserve the size of the feature maps as otherwise, they shrink at each layer which is not desirable.

ACM AI

CNN: Visualization of Stride and Padding







Stride

- Examples show a stride of 1 vs a stride of 2
- With a bigger stride, there is less overlap between the receptive fields. This makes the resulting feature map smaller since we're skipping over potential locations
- We can see the feature map is smaller than the input because the filter needs to be contained in the input
- If we don't want that, we use padding

Padding

- Refers to surrounding the input with zeroes or the values on the edge to maintain dimensionality (preserve the size of the feature maps as otherwise, they shrink at each layer which is not desirable.

ACM AI

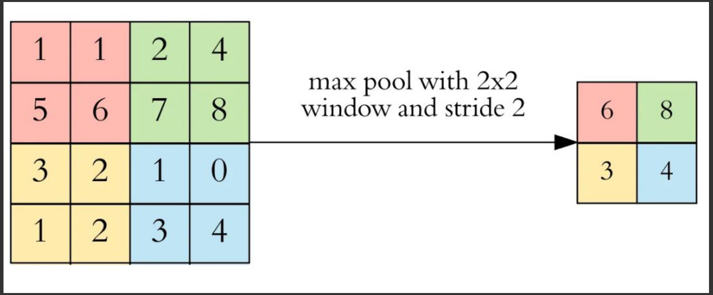
CNN: Pooling Layer

The intuitive reasoning behind pooling is that once we know that a specific feature is in the original input, it's exact location is not as important as its relative location to the other features.

- After convolution, pooling is performed to reduce the dimensionality. This enables us to reduce the number of parameters -> shortens training time and combats overfitting
- Pooling downsample each feature map independently to reduce the height and width while keeping depth intact

Types of Pooling

- The most common is max pooling, which just takes the max value in the pooling window. Contrary to convolution operation, pooling has no parameters.
- It slides a window over its input and simply takes the max value in the window. Similar to a convolution, we specify the window size and stride
- Pooling is typically performed with 2x2 windows, stride 2, and no padding. Compared to convolution which is done with 3x3 windows, stride 1, and with padding



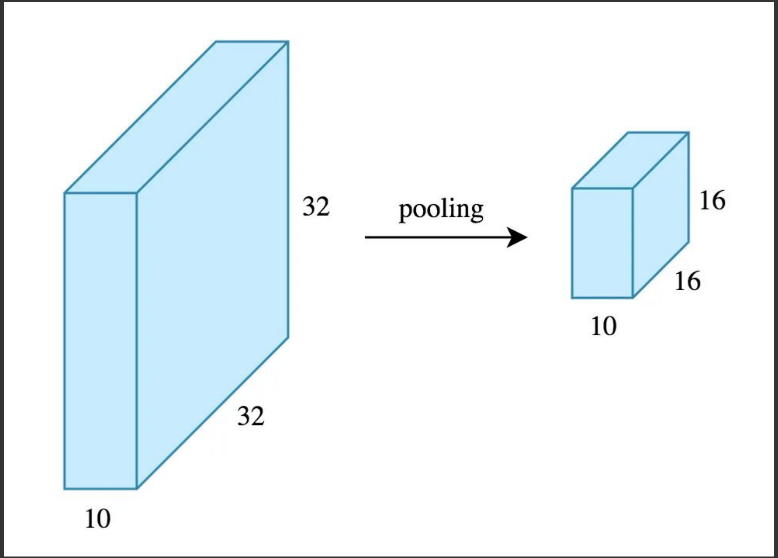
- Note that the window and stride configuration halves the size of the feature map. This is the main case of pooling, downsampling the feature map while keeping the important information

ACM AI

CNN: Max Pooling Visualization

Feature Dimensions before and after pooling

- input to the pooling layer has the dimensionality 32x32x10
- Pooling parameters with max pooling set at 2x2 window, stride 2, and no padding
- The result is a 16x16x10 feature map
- The height and width of the feature map are halved, but the depth doesn't change because pooling works independently on each depth
- By halving the height and the width, we reduce the number of weights to ¼ of the input.
- Considering that we typically deal with millions of weight in CNN architecture, this reduction is very significant



ACM AI

CNN: Fully Connected Layer

- Fully connected layer just refers to neural networks
- After the convolution and pooling layers, fully connected layers are added to complete the CNN architecture
- The output of the convolution and pooling layers are 3D volumes so we flatten the final output of the pooling layer.
  - Flattening is simply arranging the 3D volume of numbers into a 1D vector
- This ensures it becomes a vector and can therefore, be taken in as an input by the fully connected layer.

Training

- Now we know the entire CNN architecture! What about training?
- The CNN is trained the same way neural networks are trained, which is backpropagation with gradient descent.

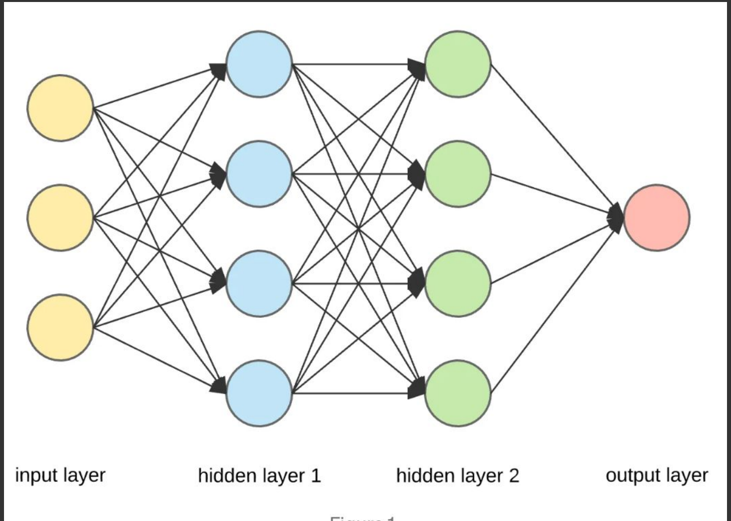


Figure 1

ACM AI

CNN: Optimizing the Architecture

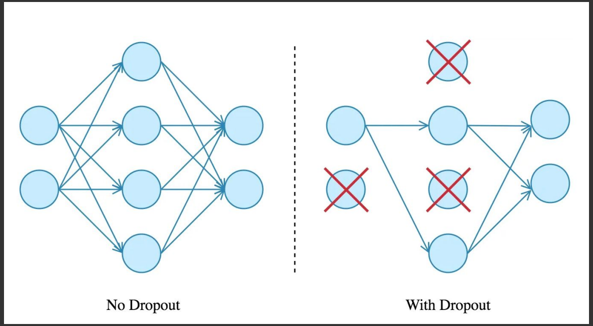
While the main three layers have been introduced, here's a couple techniques added for the purpose of optimization

Activation Functions

- Used to learn and approximate any kind of continuous and complex relationship between variables of the network.
- Simply, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network.
- It decides whether the input to the work is important or not to predict
- It adds non-linearity to the network.
- Ex: ReLu, Softmax, tanH, and Sigmoid are common activation functions, each with distinct uses

Dropout

- When features are all connected in the fully connected layer, it can cause overfitting in the training data.
- A dropout layer is utilized where a few neurons are dropped from the neural network during training process (to reduce the size of the model)
- A dropout rate of 0.5 will drop 50% of the neurons/nodes randomly
- This improves the performance as it prevents overfitting by making the network more simple and generalizable



CNN: Intuition

That was a lot of information…ok,here’s the big picture intuition.

- A CNN model can be thought of as a combination of two components: feature extraction and classification.
- The convolution and pooling layers perform feature extraction.
    - Ex: Given an corgi image, the convolution layer detects features such as two eyes, long ears, four legs, a short tail, and so worth.
- The fully connected layers then act as a classifier on top of these features, and assign a probability for the input image being a dog.

Once again, so why is CNN impressive?

- Automatically detecting meaningful features given only an image and a label is not an easy task. The convolution layers learn such complex features by building on top of each other. The first layers detect edges, the next layers combine them to detect shapes, then the following layers merge these information to infer that this feature is say, a nose.
- The CNN doesn’t know what a nose is, but by seeing a lot of them in a image, it learns to detect that as a feature.
- The fully connected layers then learn how to use these features to correctly classify the images

ACM AI

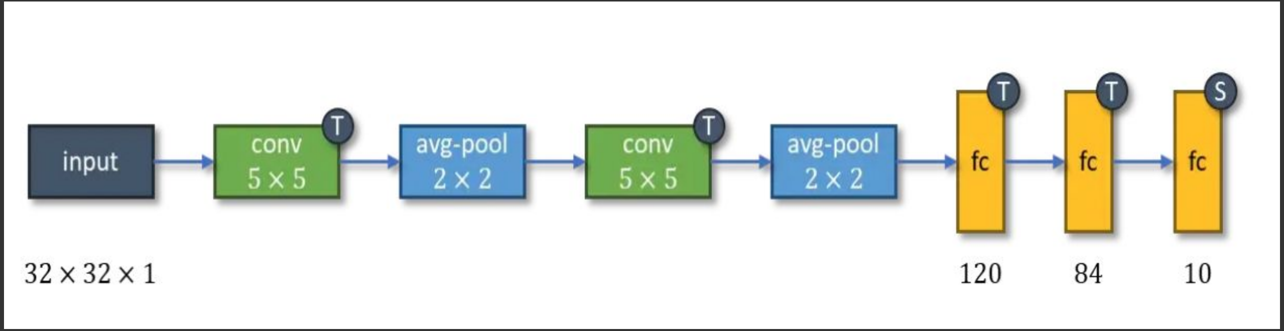
Examples of CNN

What are some of the best CNN models?

ai ACM AI

LeNet-5: The Origin of the CNN Era

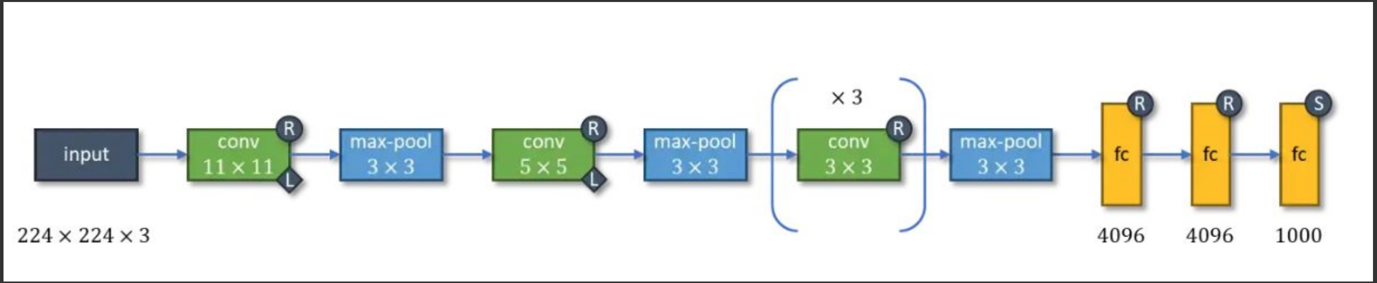
- The **LeNet** was introduced in 1998 in a paper by a number of authors who have now, all became significant figures in AI research (LeCun
- Sparked the possibility of deep neural networks in practice, but was difficult to implement due to limited computation and memory capability until the 2010s
- Made up of 7 layers: 3 convolutional layers, 2 pooling layers, and 2 fully connected layers
- 60k parameters



ACM AI

AlexNet: Local Response Normalization

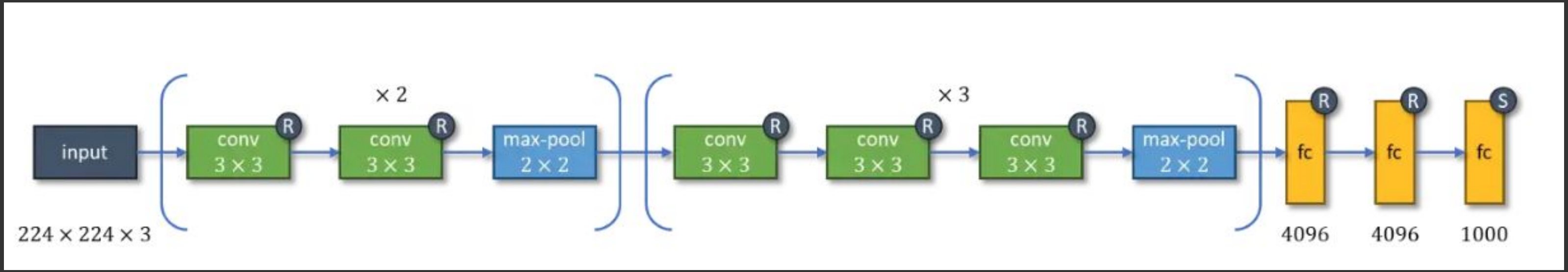
- AlexNet performed so well on the historically complex ImageNet dataset that it lead to the development of other CNNs - VGG, Inception, ResNet, and EfficientNet
- Introduced the ReLU activation function and LRN
- ReLU became so popular that almost all CNN architectures developed after AlexNet used ReLU in their hidden layers (compared to the use of tanH activation functions in LeNet-5)
- Consists of 8 layers - 5 convolution layers and 3 fully connected layers
- Can be trained with 60 million parameters



ACM AI

VGG-16: Deep Layers

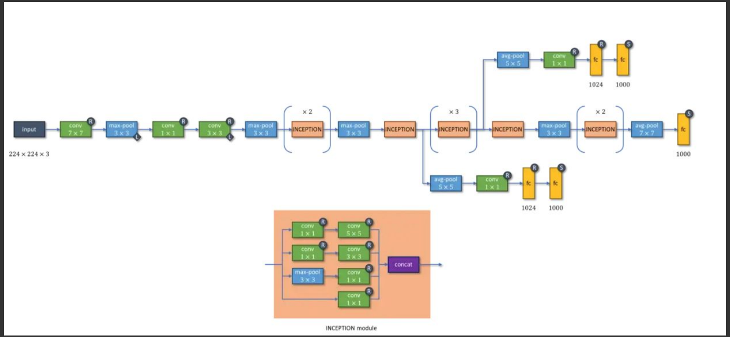
- The VGG families of CNN models came to be by pushing the depth to 11-19 layers. VGG-16 has 16 layers
- 13 convolution layers plus 3 fully connected layers
- 138 million parameters



ACM AI

Inception-v1: Inception Module

- There's a caveat to having many layers: exploding/vanishing gradients
- The exploding gradient is a problem when large error gradients accumulate and result in unstable weight updates during training
- The vanishing gradient is a problem when the partial derivative of the loss functions approaches a value close to zero and the network couldn't train
- Inception tackles this issue by adding two auxiliary classifiers connected to intermediate layers - this is done to increase the gradient signal that gets propagated back.
- Introduces the **inception module**, four series of one or two convolution and max-pool layers stacked in parallel and concatenated in the end. Aims to approximate an optimal local sparse structure in a CNN by allowing the uses of multiple types of kernel sizes instead of being restricted to single kernel size

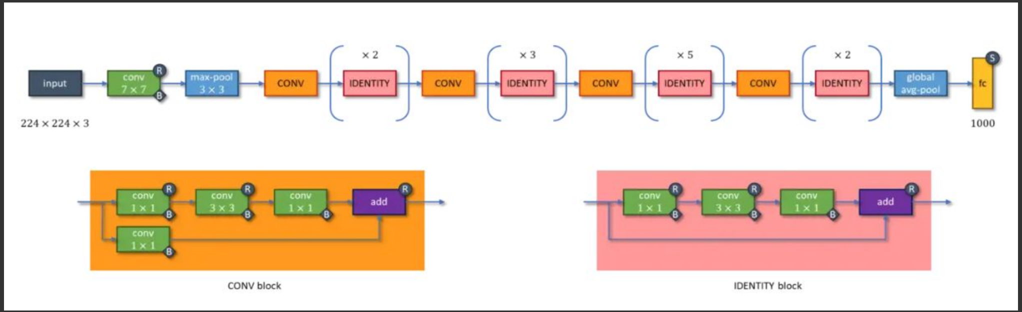


- 22 layers: 3 convolution, 18 layers of 9 inception modules (each module is two layers), and 1 fully connected layer

ACM AI

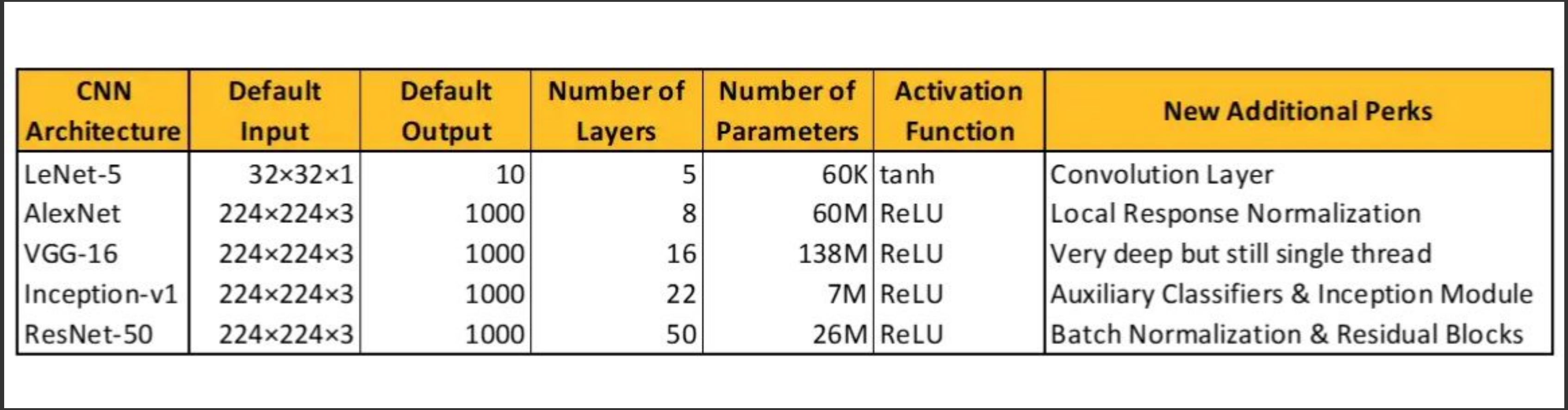
ResNet-50: Residual Blocks

- Degradation problem: with network depth increasing, accuracy gets saturated and then degrades rapidly
- Introduces **bottleneck residual blocks**, there are two types
  - Identity Blocks: consists of 3 convolution layers
  - Convolution Blocks: same as identity block but the input is first passed through a convolution layer before being added to the last convolution layer of the main series
- 50 layers total
  - 9 layers of 1 convolution block and 2 identity blocks + 12 layers of 1 convolution block and 3 identity blocks + 18 layers of 1 convolution block and 5 identity blocks + 9 layers of 1 convolution block and 2 identity blocks + 1 fully connected layer



ACM AI

Summary of all the CNNs



CNN Architecture	Default Input	Default Output	Number of Layers	Number of Parameters	Activation Function	New Additional Perks
LeNet-5	32×32×1	10	5	60K	tanh	Convolution Layer
AlexNet	224×224×3	1000	8	60M	ReLU	Local Response Normalization
VGG-16	224×224×3	1000	16	138M	ReLU	Very deep but still single thread
Inception-v1	224×224×3	1000	22	7M	ReLU	Auxiliary Classifiers & Inception Module
ResNet-50	224×224×3	1000	50	26M	ReLU	Batch Normalization & Residual Blocks

ACM AI

Interactive Notebook

https://acmurl.com/cv2-interactive

ACM AI

Thanks for attending!

Make sure to check in to get membership points

Intro to CV: Convolutional Neural Ne...

Check-in code: cnns