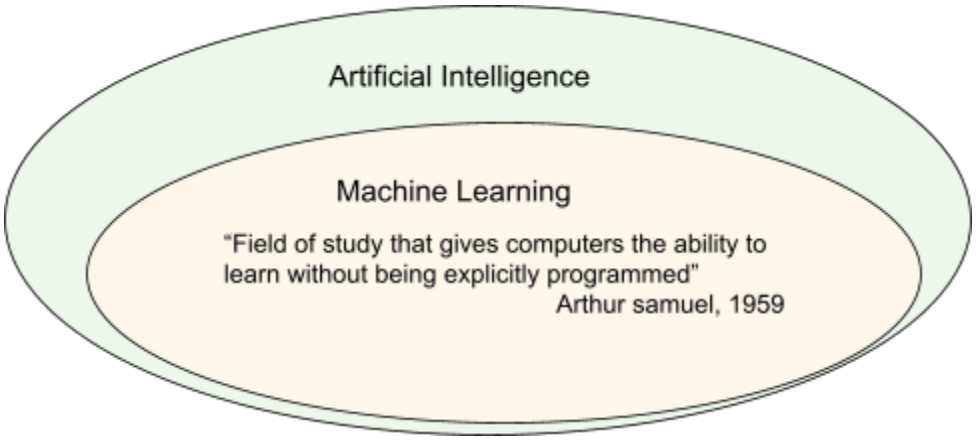
Machine Learning Guide

1. Introduction

Machine Learning is when a computer acquires the ability to solve a task through examples, rather than human-encoded handcrafted rules. (Deep Learning with PyTorch, Eli stevens). A general class of algorithms able to approximate, complicated, non-linear processes extremely effectively. Alan M. Turing thought about . . . the question of whether Machines Can Think,

a question . . . about as relevant as the question of whether Submarines Can Swim. Deep learning leverages large amounts of data to approximate complex functions whose inputs and outputs are far apart. (Deep Learning with PyTorch, Eli stevens).



There are tens of thousands of machine learning algorithms with hundreds of algorithms being added to the repertoire every year. Every machine learning algorithm has three components; Representation, Optimization and Evaluation.

Epoch - went through all the dataset once

Iteration - went through the batch once

Batch normalization - is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch.

https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

Smaller value of neurons on deep neural networks layers helps.

Why is deep learning taking off?

Scale drives deep learning progress

- Data
- Computation
- Algorithms

2. Machine Learning in Practice

ML in practice

- Understand domain, prior knowledge and goals

- Data integration, selection, cleaning, pre-processing, etc.

- Learn models

- Interpret results

- Consolidate and deploy discovered knowledge

3. Regression

The simplest machine learning model.

3.1 Linear Regression

Output: Continuous

Example: housing price prediction

3.2 Logistic Regression

When we want our output to be continuous, we use logistic regression instead of linear regression.

Output: between 0 and 1

Ex. we want probability y_hat(y=1|x)

4. Support vector machines

5. KNN (K-Nearest Neighborhood)

2. Feature Scaling

5. Clustering K-means

6. Classification

6.1 Support Vector Machine (SVM)

6.2 Binary Classification

K nearest neighborhood: is a simple but powerful classification algorithm. It is classified based on similarity measures.

7. Supervised Learning

Online advertising

Computer vision

Speech recognition

		Application	Architecture

Structured data - each of the features has well-defined meaning

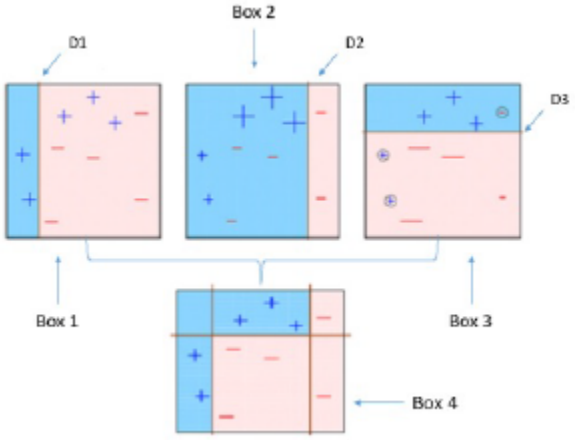
Unstructured data - audio, images, text

People are good at understanding unstructured data

COURSERA

XGBoost

XGBoost stands for e**X**treme **G**radient **Boost**ing. XGBoost is a powerful iterative learning algorithm based on gradient boosting. XGBoost is robust and highly versatile, with custom objective loss function compatibility.



How does XGBoost work?

Tree-based Boosting algorithm.

4 critical parameters for tuning.:

- etha: ETA or “Learning Rate”
- Max_depth: Controls the “height” of the tree via splits.
- gamma: Minimum required loss for the model to justify a split.
- Lambda: L2(Ridge) regularization on variable weights.

Why use XGBoost?

- Scalable

- Won many kaggle competitions

- All of the advantages of gradient boosting

- Frequent Kaggle data competition champion

- Utilizes CPU Parallel Processing by default.

- Two main reasons for use: Low Runtime and High Model Performance

8. Computational Graphs

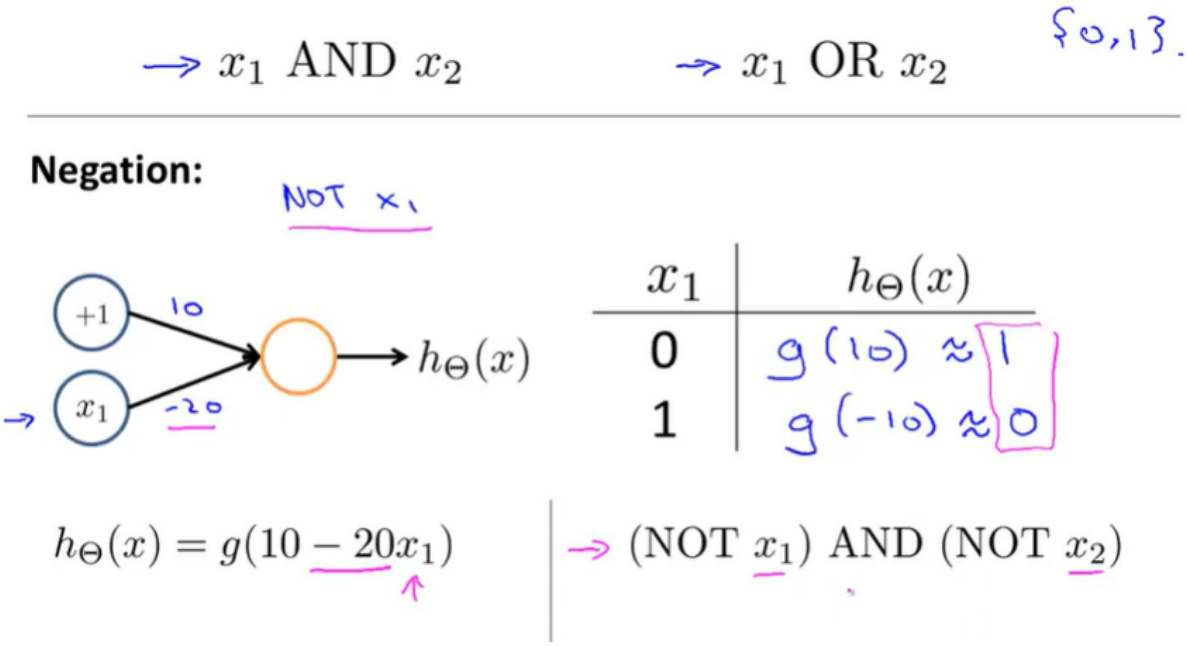
Computational graph is a way to represent a math function in the language of a graph theory[1].

[1] Graph theory is a branch of mathematics concerned with a network of points connected by lines.

9. Neural Networks

Logic Gates

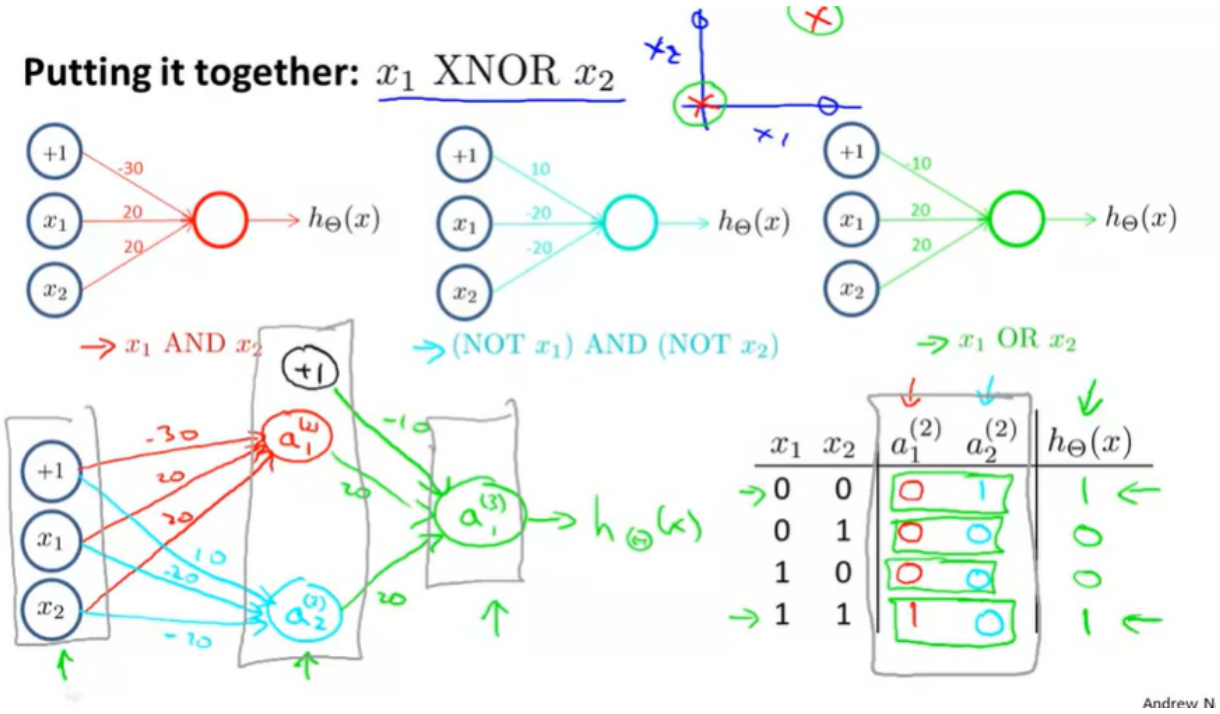
Negation



XOR

Building XOR (non-linear decision boundary)

To build an XOR function, you need a hidden layer to increase the complexity.



And this is a sort of intuition about why neural networks can compute pretty complicated functions. That when you have multiple layers you have a relatively simple function of the inputs of the second layer. But the third layer I can build on that to complete even more complex functions, and then the layer after that can compute even more complex functions.

10. Convolutional Neural Networks

11. Sequence Models

Recurrent Neural Network (RNN)

Long short-term Memory (LSTM)

10. Performance Metrics

Performance metrics are a part of every machine learning pipeline. They tell you if you're making progress, and put a number on it. All machine learning models, whether it's linear regression or a SOTA technique BERT[2], need a metric to judge performance.

11. Neural Network Evaluation

Learning rate

12. Semantic Segmentation

Objectives:

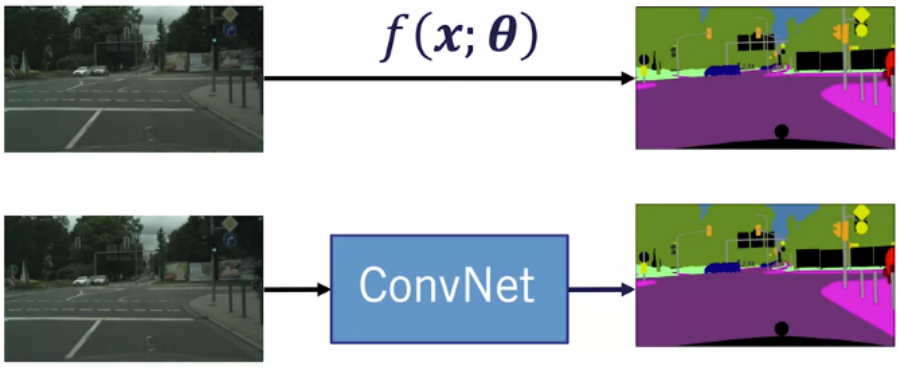
- Learn how to use a convolution neural network to perform semantic segmentation
- Learn the different layers required for the good performance of semantic segmentation models

[2] BERT stands for Bidirectional Encoder Representations from Transformers. It is a transformer based model by Google that provides a pre-trained model for Natural Language Processing (NLP).

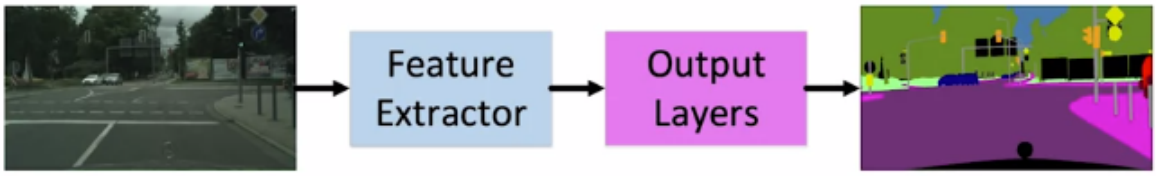
What is semantic segmentation? Semantic segmentation takes a camera image input and provides a category classification. Using the convolutional net for semantic segmentation is actually a little easier compared to using them for object detection. Unlike ConvNets for object detection, the training and inference stages[3] are practically the same for semantic segmentation.

As we said, semantic segmentation takes a camera image input and provides a classification for every pixel in that image as output. This problem can be modeled as a function approximation problem. ConvNets can be used to approximate the required function.

[3] What is an inference stage?



What kind of ConvNet model can we use to perform this function approximation? One idea is to use the same ConvNet model we used for object detection. This is a feature extractor followed by an output layer. We don’t need anchors here, as we aren’t trying to localize objects.



Let’s see how effective this model is for performing semantic segmentation. Let's start by analyzing the feature extractor. As with the feature extractor used for object detections, we can use the VGG architecture. Let’s see what happens to the size of an MXNx3 input image, as we go through this feature extractor.

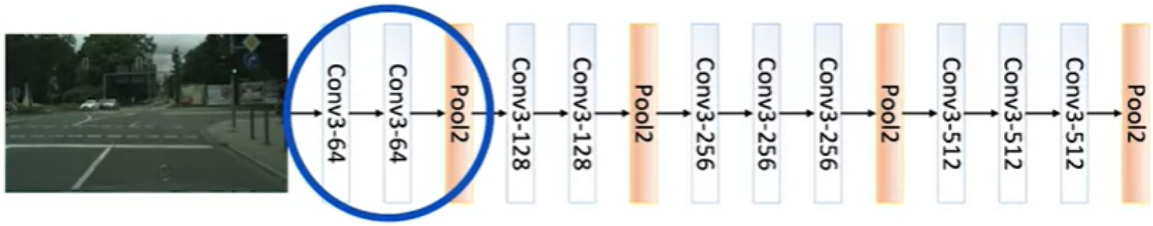
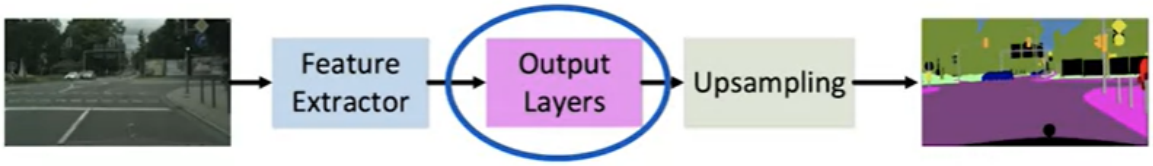


	Image	Conv1
Width	M	M/2
Height	N	N/2
Depth	3	64

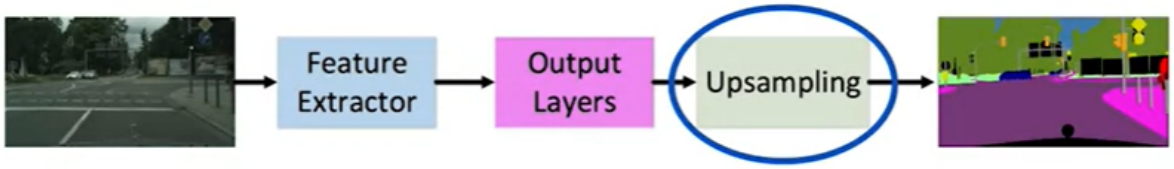
As with object detection , our resolution will be reduced by half after every pooling layer.

	Image	Conv1	Conv2	Conv3	Conv4
Width	M	M/2	M/4	M/8	M/16
Height	N	N/2	N/4	N/8	N/16
Depth	3	64	128	256	512

The input image is reduced 16 times, so how can we do categorization for every pixel? A simple solution would be, we can first compute the output by passing the 16 times downsampled output feature map through a softmax layer.



We can then determine the class of every pixel in the subsampled output by taking the highest class score obtained by the softmax layer. The final step would be to proceed to upsample the downsampled output back to the original image resolution.



To understand why naive upsampling might produce inadequate results?

13. Generative Models

13.1 Deep Generative Model

Why generative models?

13.2 Generative Adversarial Network (GAN)

GAN is an approach to generative modeling using deep learning methods. Generative modeling is an unsupervised learning task in machine learning that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset.

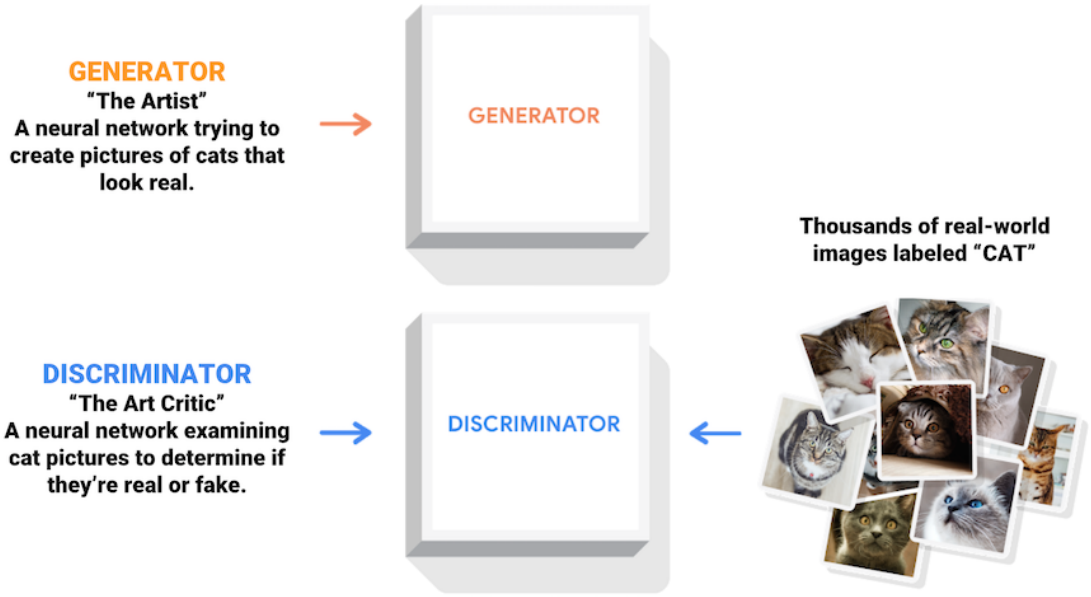
GANs are a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models: the generator model that we train to generate new examples, and the discriminator model that tries to classify examples as either real (from the domain) or fake (generated). The two models are trained together in a zero-sum game, adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples.

Deep Convolutional Generative Adversarial Network

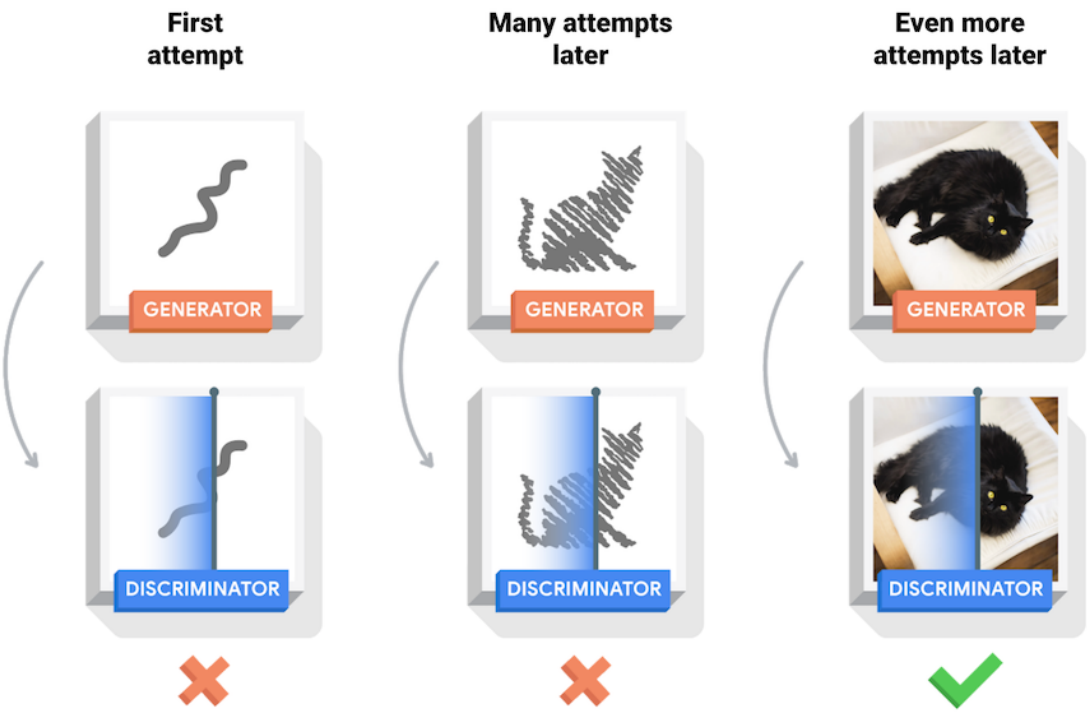
https://www.tensorflow.org/tutorials/generative/dcgan

What are GANs?

Generative Adversarial Networks (GANs) are one of the most interesting ideas in computer science today. Two models are trained simultaneously by an adversarial process. A generator ("the artist") learns to create images that look real, while a discriminator ("the art critic") learns to tell real images apart from fakes.



During training, the *generator* progressively becomes better at creating images that look real, while the *discriminator* becomes better at telling them apart. The process reaches equilibrium when the *discriminator* can no longer distinguish real images from fakes.



This notebook demonstrates this process on the MNIST dataset. The following animation shows a series of images produced by the *generator* as it was trained for 50 epochs. The images begin as random noise, and increasingly resemble handwritten digits over time.

Learn more about GAN: http://introtodeeplearning.com/

Super-resolution

ESRGAN (Enhanced Super-Resolution Generative Adversarial Network)

BSRGAN (Blind Super-Resolution Generative Adversarial Network)

13.3 Pix2Pix

13.4 cGAN

Conditional GAN

Discriminator loss = real loss + fake loss

real_loss

fake_loss

Generator loss = real_fake_loss + var_1*gt_loss (ground_truth, predicted) + var_2*wf_loss (ground_truth, predicted)

real_fake_loss

Gt_loss = norm (predicted - ground_truth)

wf_loss =

cGAN loss function

ref

https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/

14. Multimodal Machine Learning

Objectives:

- Introduction
- Representation Fusion
- Alignment
- VilBERT
- OpenAI - CLIP

FOUNDATIONS & RECENT TRENDS IN MULTIMODAL MACHINE LEARNING: PRINCIPLES, CHALLENGES, & OPEN QUESTIONS

Paul Pu Liang, Amir Zadeh, Louis-Philippe Morency
Language Technologies Institute & Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
{pliang,abagherz,morency}@cs.cmu.edu

Abstract

Multimodal machine learning is a vibrant multi-disciplinary research field that aims to design computer agents with intelligent capabilities such as understanding, reasoning, and learning through integrating multiple communicative modalities, including linguistic, acoustic, visual, tactile, and physiological messages. With the recent interest in video understanding, embodied autonomous agents, text-to-image generation, and multisensor fusion in application domains such as healthcare and robotics, multimodal machine learning has brought unique computational and theoretical challenges to the machine learning community given the heterogeneity of data sources and the interconnections often found between modalities. However, the breadth of progress in multimodal research has made it difficult to identify the common themes and open questions in the field. By synthesizing a broad range of application domains and theoretical frameworks from both historical and recent perspectives, this paper is designed to provide an overview of the computational and theoretical foundations of multimodal machine learning. We start by defining two key principles of modality *heterogeneity* and *interconnections* that have driven subsequent innovations, and propose a taxonomy of 6 core technical challenges: *representation*, *alignment*, *reasoning*, *generation*, *transference*, and *quantification* covering historical and recent trends. Recent technical achievements will be presented through the lens of this taxonomy, allowing researchers to understand the similarities and differences across new approaches. We end by motivating several open problems for future research as identified by our taxonomy.

15. Transformers

16. Natural Language Processing

16.1 Introduction

16.2 Python Text Basics

We will cover the basics of python's text capabilities. Opening text files, working with pdf files and regular expressions.

16.3 NLP Python Basics

We will use specialized NLP libraries such as Spacy and NLTK. The two most popular NLP libraries for python. Using specialized NLP libraries such as Spacy and NLTK. The two most popular NLP libraries for python. And within this section we'll discuss things like spacy pipelines, tokenizations, stemming, lemmatization, stop words, and vocabulary and phrase matching.

16.4 Parts of Speech Tagging

16.5 Text Classification

16.5 Semantics and Sentiment Analysis

16.6 Topic Modeling

16.7 ChatBots and Advanced Material

A deep learning model that adopts the mechanism of self-attention, differentially weighting the significance of each part of the input data.

[1] https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)

SwinIR (Swin Image Restoration)

Coefficient Matrix

https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative

An Aesop's Fable: The Boy Who Cried Wolf (*compressed*)

A shepherd boy gets bored tending the town's flock. To have some fun, he cries out, "Wolf!" even though no wolf is in sight. The villagers run to protect the flock, but then get really mad when they realize the boy was playing a joke on them.

[Iterate previous paragraph *N* times.]

One night, the shepherd boy sees a real wolf approaching the flock and calls out, "Wolf!" The villagers refuse to be fooled again and stay in their houses. The hungry wolf turns the flock into lamb chops. The town goes hungry. Panic ensues.

Let's make the following definitions:

- "Wolf" is a **positive class**.
- "No wolf" is a **negative class**.

We can summarize our "wolf-prediction" model using a 2x2 confusion matrix that depicts all four possible outcomes:

True Positive (TP): • Reality: A wolf threatened. • Shepherd said: "Wolf." • Outcome: Shepherd is a hero.	**False Positive (FP):** • Reality: No wolf threatened. • Shepherd said: "Wolf." • Outcome: Villagers are angry at shepherd for waking them up.
False Negative (FN): • Reality: A wolf threatened. • Shepherd said: "No wolf." • Outcome: The wolf ate all the sheep.	**True Negative (TN):** • Reality: No wolf threatened. • Shepherd said: "No wolf." • Outcome: Everyone is fine.

Dice coefficient

The dice coefficient can be used to compare the pixel-wise agreement between a predicted segmentation and its corresponding ground truth. Given by the following formula.

$$\frac{2 * |X \cap Y|}{|X| + |Y|}$$

17. Text to Image Generation

Contrastive models like CLIP [i-1] have been shown to learn robust representations of images that capture both semantics and style. To leverage these representations for image generation, we propose a two-stage model: a prior that generates a CLIP image embedding given a text caption, and a decoder that generates an image conditioned on the image embedding.

https://arxiv.org/abs/2204.06125

[i-1] CLIP (Contrastive Language-Image Pre-Training)

Figure x, Random samples from unCLIP for prompt “A close up of a hand palm with leaves growing from

it.”

OpenAI’s DALL-E 2 - AI-Based Art

18. TensorFlow

TensorFlow hub:

is a repository of trained machine learning models.TensorFlow Hub is a repository of trained machine learning models ready for fine-tuning and deployable anywhere. Reuse trained models like BERT and Faster R-CNN with just a few lines of code.

```
🎯!pip install --upgrade tensorflow_hub
```

https://www.tensorflow.org/hub

19, Object Detectors

An Effective and Efficient Implementation of Object Detector

https://crossminds.ai/video/pp-yolo-an-effective-and-efficient-implementation-of-object-detector-60787479ed1608afe13a6f4b/

20. Transfer Learning

21. Resources

OpenAI

Perceptilabs

https://www.perceptilabs.com/papers

22. Performance Metrics

Introduction

Performance metrics are a part of every machine learning pipeline. They tell you if you're making progress, and put a number on it. All machine learning models, whether it's linear regression or a SOTA technique BERT, need a metric

For Regression

For Classification

		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

Accuracy = TP + TN / TP+TN +FP+FN

Precision = TP / TP + FP = True positive/total predicted positive

Recall = TP / TP + FN = TP/ Total actual positive

F1-Score

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Reference

https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd

https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Others

23. Modern Practices

A deep network can represent functions of increasing complexity by adding more layers and more units in a layer. Given sufficiently big models and sufficiently extensive datasets of labeled training samples, deep learning can be used to complete most tasks that involve quickly mapping an input vector to an output vector. For the time being, deep learning cannot be used to solve problems that cannot be solved by simply associating one vector to another or those are so challenging that solving them would take some thought and reflection.

Glossary

Loss function: The loss function is a function that will need to define to measure how good our output y hat is when the true label is y.