

Deep Learning

Week 1 : Class Overview &
The Fundamentals (Part 1):
The Neuron, Multi-layer Perceptron,
Optimization, Back-Prop



Class Overview : Instructors



TBD

Arturo Deza

TRANSFORMATEC

Main Instructor

Class TA



(Introductions around the class)

What is your name?

What is your major (*Especialidad*)?

Why are you taking this class?

What do you want to learn in this class?

	Monday (Theory; Deza)	Wednesday (Lab; Deza/TA)	Friday (Lab; Deza + TA Guest)
Focus : Vision	Week 1 Overview of Basics of Deep Learning (MLP + Optimization + BackProp)	Overview of Basics of Deep Learning	Assigned In-Class Reading & Discussion + Implementation
	Week 2 CNNs (Convolution, RELU, Pooling, De-Convolution, Loss) - Part 1	CNNs - Code Review	Assigned In-Class Reading & Discussion + Implementation
	Week 3 CNNs (Convolution, RELU, Pooling, De-Convolution, Loss) - Part 2	CNNs - Code Review	First Quiz
	Week 4 GANs (Nash Equilibria, Loss, Resolution)	GANs (Nash Equilibria, Loss, Resolution) - Code Review	Lab 1 (Image Classification; Kaggle)
	Week 5 VAEs (Stochastic Sampling, KL-Divergence)	VAEs (Stochastic Sampling, KL-Divergence) - Code Review	Assigned In-Class Reading & Discussion + Implementation
	Week 6 Signal Processing + Image Compression + Rate-Distortion Theory	Image Compression - Code Review	Lab 2 (Generative AI; Kaggle)
	Week 7 Presentation Students 1A	Presentation Students 1B	Guest Invited Lecture
	Week 8 Sparse Coding (Theory + Code + Neuroscience)	Course Review	MidTerm Exam
Focus: Frontier Topics + DuckieBot Focus	Week 9 NeuroAI: Experimental Design + Representational Similarity Analysis	Assigned In-Class Reading & Discussion + Implementation	DuckieBot Engineering (Work-In-Class)
	Week 10 Advanced Sequence Models: RNNs, LSTMs, Seq2Seq, Transformers - 1	Assigned In-Class Reading & Discussion + Implementation	DuckieBot Engineering (Work-In-Class)
	Week 11 Advanced Sequence Models: RNNs, LSTMs, Seq2Seq, Transformers - 2	DuckieBot Engineering (Work-In-Class)	Lab 3 : DuckieBot Progress (Obstacle Route Basic)
	Week 12 Advanced RL : Large Language Models, Games, Protein Folding	Assigned In-Class Reading & Discussion + Implementation	DuckieBot Engineering (Work-In-Class)
	Week 13 Autonomous Driving - 1	Assigned In-Class Reading & Discussion + Implementation	DuckieBot Engineering (Work-In-Class)
	Week 14 Presentation Students 2B	Presentation Students 2A	Guest Invited Lecture
	Week 15 Autonomous Driving - 2	DuckieBot Engineering (Work-In-Class)	Lab 4 : DuckieBot Final Track
	Week 16 Diffusion Models	Course Review	Final Exam

Ponderacion de Evaluaciones Finales

First Test	10 Nota Individual
Mid-Term	10 Nota Individual
Final Exam	10 Nota Individual
Lab 1	10 Nota Individual
Lab 2	10 Nota Individual
Lab 3	10 Nota Individual
Lab 4	10 Nota Individual
Presentation 1	15 Nota Grupal
Presentation 2	15 Nota Grupal
Total	100



Ojo : Esta clase sí tendrá modo candado!



The screenshot shows the ChatGPT interface on a web browser. The title bar includes tabs for "SpecDecoder | Mobile Phone Pri", "Paraphrasing Tool - QuillBot AI", and "New chat". The URL in the address bar is <https://chat.openai.com/chat>. The main header features the "SpecDecoder" logo and "ChatGPT". On the left sidebar, there are several "New chat" options listed, along with "Clear conversations", "Upgrade to Plus" (marked as NEW), "Dark mode", "Improve ChatGPT", "Updates & FAQ", and "Log out". The main content area is titled "ChatGPT" and contains three sections: "Examples", "Capabilities", and "Limitations". Each section displays a few examples of interactions with ChatGPT, such as explaining quantum computing, generating creative ideas for a birthday, making an HTTP request in Javascript, and declining inappropriate requests. The "Limitations" section notes that ChatGPT may occasionally generate incorrect information, produce harmful instructions, or have limited knowledge of world events after 2021.

Examples	Capabilities	Limitations
"Explain quantum computing in simple terms" →	Remembers what user said earlier in the conversation	May occasionally generate incorrect information
"Got any creative ideas for a 10 year old's birthday?" →	Allows user to provide follow-up corrections	May occasionally produce harmful instructions or biased content
"How do I make an HTTP request in Javascript?" →	Trained to decline inappropriate requests	Limited knowledge of world and events after 2021

ChatGPT Feb 13 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

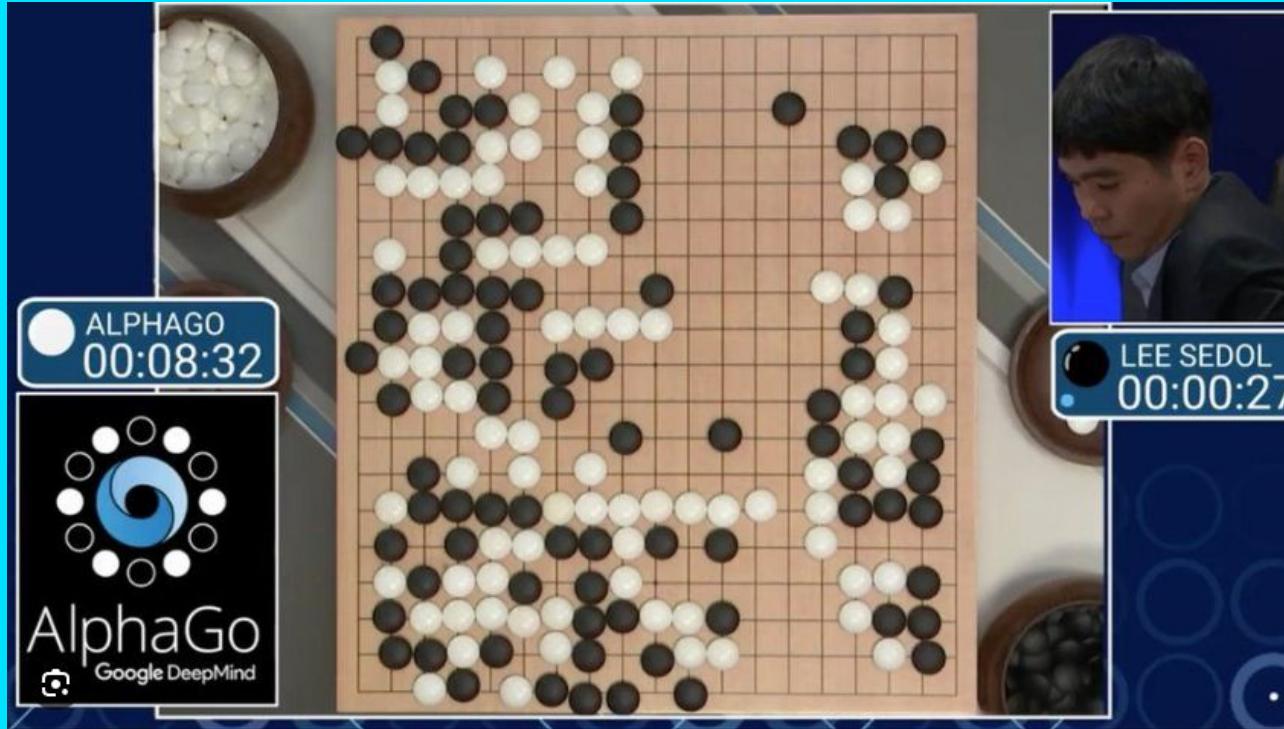
¿Qué es la Inteligencia Artificial?



¿Qué es la Inteligencia Artificial?



¿Qué es la Inteligencia Artificial?



¿Qué es la Inteligencia Artificial?



¿Qué es la Inteligencia Artificial?

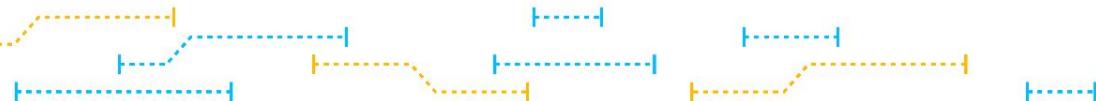


Problemas Modernos de AI

- **Expensive:** Millones de Dólares para Entrenar una Red Neuronal con una SuperComputadora.
- **Non-Generalizable:** Se necesita billones de data-points para entrenar a una computadora para hacer una tarea precisa.
- **Non-Robust:** En varias ocasiones los sistemas fallan ante pequeñas distorsiones en la entrada del sistema



What is Learning?



What is the Difference between Machine Learning & Deep Learning?

 perplexity

New Thread Ctrl + I

Q Home
D Discover
L Library
S Sign in

Sign Up

Try Pro
Upgrade for image upload,
smarter AI, and more Pro

Anonymous

What is the difference between Machine Learning & Deep Learning?

Sources

Machine Learning vs Deep Learning - Data...

 data science.stackexchange · 1

Difference Between Machine Learning and...

 geeksforgeeks · 2

Deep Learning vs Machine Learning - ...

 aws.amazon · 3

 View 2 more

Answer

The main difference between Machine Learning (ML) and Deep Learning (DL) lies in how they handle data and learn patterns. Machine Learning involves training algorithms to identify patterns and relationships in data, while Deep Learning uses neural networks with multiple layers to analyze complex patterns and relationships. In simpler terms, Deep Learning is a subset of Machine Learning, specifically focusing on learning with deep neural networks that have many layers. While Machine Learning requires engineers to design and predefine features manually, Deep Learning learns features automatically. Deep Learning is more complex and powerful, making it suitable for tasks like computer vision, natural language processing, and speech recognition. In summary, Deep Learning is a specialized subset of Machine Learning, which, in turn, is a subset of Artificial Intelligence [2](#) [5](#).



What is the difference between Machine Learning & Deep Learning?

YOU

Difference Between Machine Learning & Deep Learning

Machine Learning and Deep Learning are two concepts within the field of Data Science and are subsets of Artificial Intelligence (AI). While they are related, there are distinct differences between the two.

Machine Learning is a subfield of AI that focuses on the development of algorithms and statistical models that enable computers to learn and make predictions or decisions without being explicitly programmed. It involves training algorithms on large datasets to identify patterns and relationships and then using these patterns to make predictions or decisions about new data [1](#).

On the other hand, **Deep Learning** is a subset of Machine Learning that deals with algorithms inspired by the structure and function of the human brain. Deep learning algorithms can work with an enormous amount of both structured and unstructured data. The core concept of deep learning lies in artificial neural networks, which enable machines to make decisions. The major difference between deep learning and machine learning is the way data is presented to the machine. Machine learning algorithms usually require structured data, whereas deep learning networks work on multiple layers of artificial neural networks [2](#).

In summary, deep learning is a specialized subset of machine learning, which, in turn, is a subset of artificial intelligence. Deep learning algorithms are designed to process unstructured data, such as images, videos, and text, while machine learning algorithms are more suited for structured data [3](#).

It's important to note that deep learning models typically require more data points to improve accuracy compared to machine learning models, given their underlying data structure [4](#). Enterprises generally use deep learning for more complex tasks, such as virtual assistants or fraud detection [4](#).

Let me know if there's anything else I can help with!

What is the Difference between Machine Learning & Deep Learning?



ChatGPT

Machine learning and deep learning are both subsets of artificial intelligence, but they differ in terms of architecture, approach, and application. Here's a brief overview of each:

1. Machine Learning (ML):

- **Definition:** Machine learning is a branch of artificial intelligence that focuses on enabling machines to learn from data and improve their performance over time without being explicitly programmed.
- **Approach:** In traditional machine learning, algorithms are trained to perform a specific task by learning patterns and relationships from labeled or unlabeled data.
- **Techniques:** ML techniques include linear regression, logistic regression, decision trees, support vector machines (SVM), k-nearest neighbors (KNN), clustering algorithms, and more.
- **Applications:** Machine learning is widely used in various domains, including recommendation systems, spam detection, predictive maintenance, image recognition, natural language processing, and more.

2. Deep Learning (DL):

- **Definition:** Deep learning is a subset of machine learning that utilizes artificial neural networks with multiple layers (deep architectures) to learn complex representations of data.
- **Approach:** Deep learning algorithms learn hierarchical representations of data through multiple layers of interconnected nodes (neurons) in neural networks.
- **Techniques:** DL techniques include convolutional neural networks (CNNs) for image processing, recurrent neural networks (RNNs) for sequential data, generative adversarial networks (GANs) for generating new data, and more.
- **Applications:** Deep learning has achieved remarkable success in various tasks, including image classification, object detection, speech recognition, natural language understanding, autonomous driving, and many others. 

Key Differences:

- **Representation:** In machine learning, the focus is on feature engineering and manual selection of relevant features, whereas deep learning learns hierarchical feature representations from raw data.
- **Complexity:** Deep learning models are generally more complex due to the presence of multiple layers, requiring large amounts of data and computational resources for training.
- **Interpretability:** Machine learning models are often more interpretable, as the relationships between features and predictions are more transparent, whereas deep learning models are often regarded as black boxes due to their complexity.
- **Performance:** Deep learning models have shown state-of-the-art performance in various tasks, especially in domains where large amounts of data are available, but they require more data and computational resources compared to traditional machine learning approaches.

In summary, while both machine learning and deep learning aim to enable machines to learn from data, deep learning excels in learning complex patterns from large datasets and has achieved remarkable success in various fields, but it also requires significant computational resources and data to train effectively.



An example of Machine Learning vs Deep Learning





“Walking to MIT”



"Man asking Woman out for Dinner"



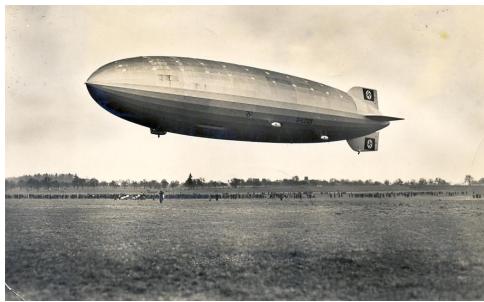


"Lima de Noche" (Lima at Night)









Marr 1982 (Vision)



Feathers or Wings?



Courtesy from Bill Freeman's
SVRHM talk @ NeurIPS 2019

Marr 1982 (Vision)

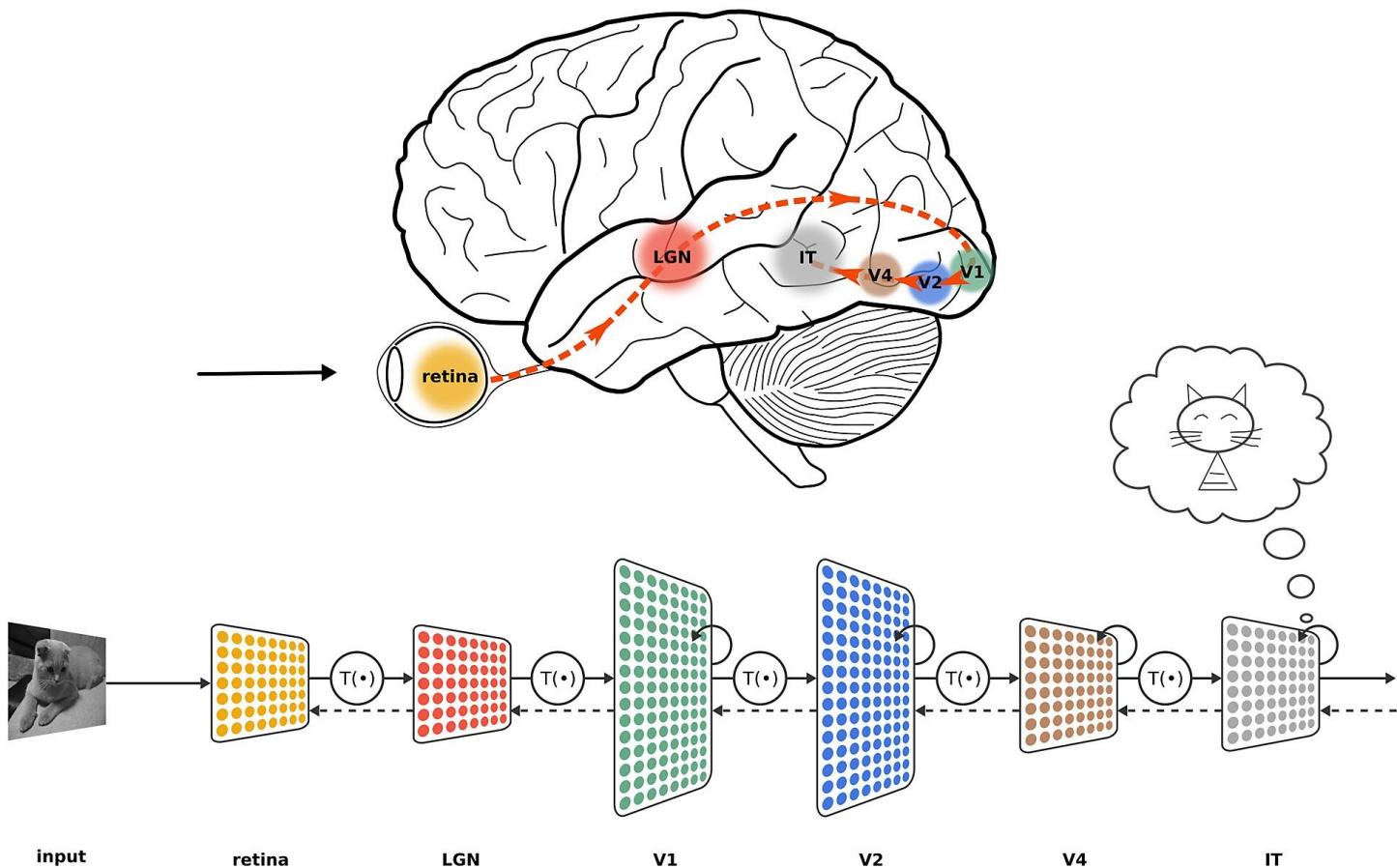
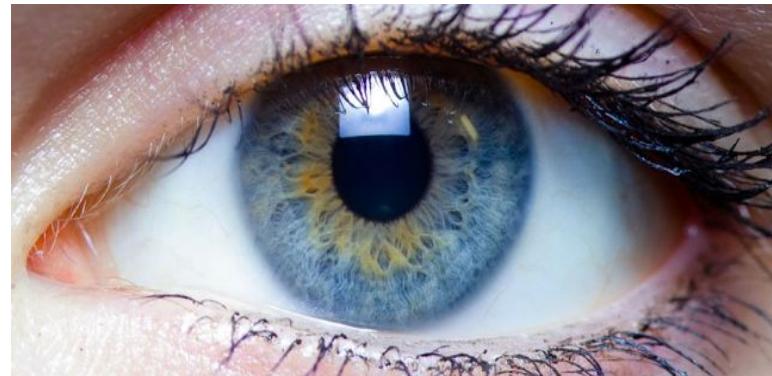
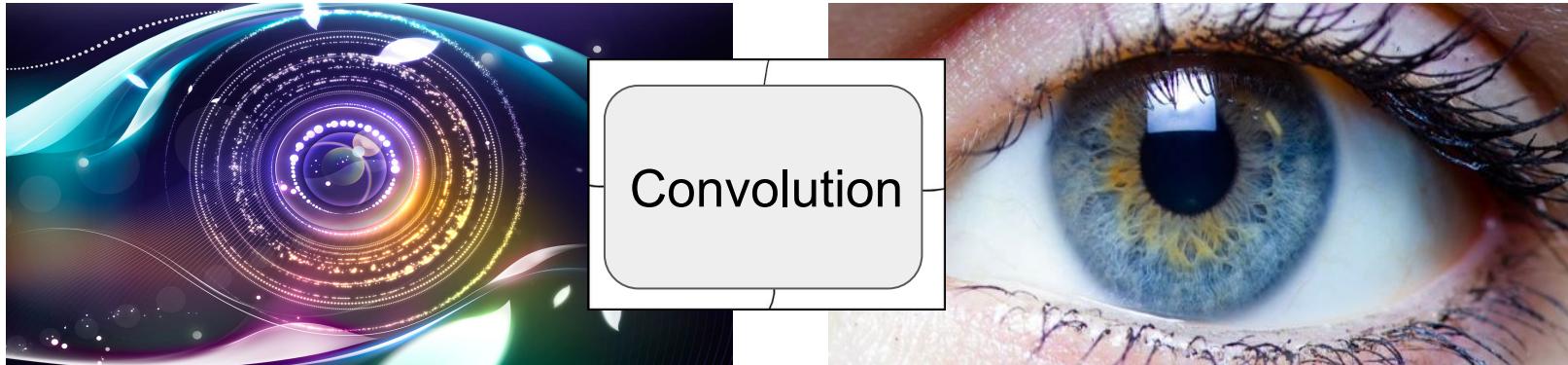
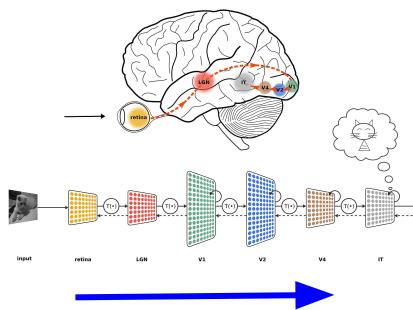
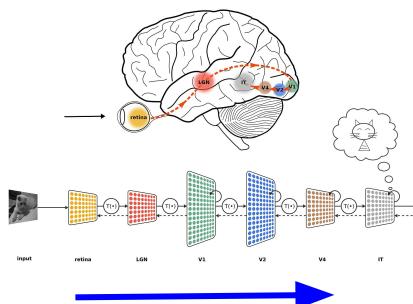
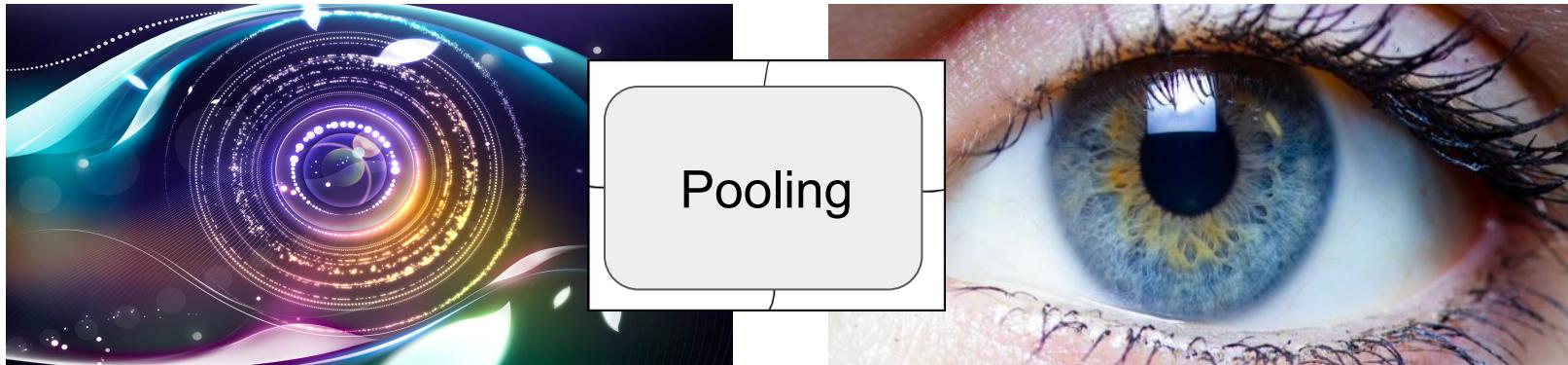
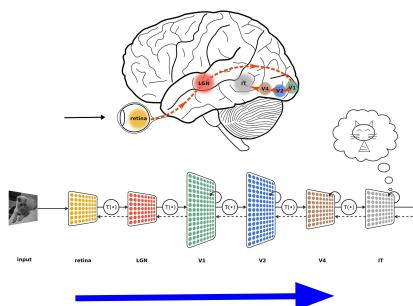


Figure: Jonas Kubilius









Important Characteristics of a 'Learning Machine'

Important Characteristics of a 'Learning Machine'

- Architecture
- Dataset
- Parameters
- Optimization / Learning Rule
- Input/Output
- Inductive Bias
- Computations

Un-Important Characteristics of a 'Learning Machine'

Un-Important Characteristics of a 'Learning Machine'

- Efficiency [Need 4 Speed]?
- Implementation of System [Recall Marr's levels | i.e. python vs C does not matter!]
- Simplicity/Complexity?

Important/UnImportant Characteristics of a Deep Learning Model



$$y = \mathbf{A}(x|\theta; \mathcal{R})$$

The Output

The Input

The Learning Rule

The Architecture

The Parameters

This diagram illustrates the components of a machine learning model. At the center is the equation $y = \mathbf{A}(x|\theta; \mathcal{R})$. Five arrows point from labels above and below the equation to specific parts of the equation:

- A vertical arrow points down to the term y , labeled "The Output".
- A vertical arrow points down to the term x , labeled "The Input".
- A vertical arrow points down to the term \mathcal{R} , labeled "The Learning Rule".
- An upward-pointing arrow originates from the label "The Architecture" and points to the symbol \mathbf{A} .
- An upward-pointing arrow originates from the label "The Parameters" and points to the term θ .

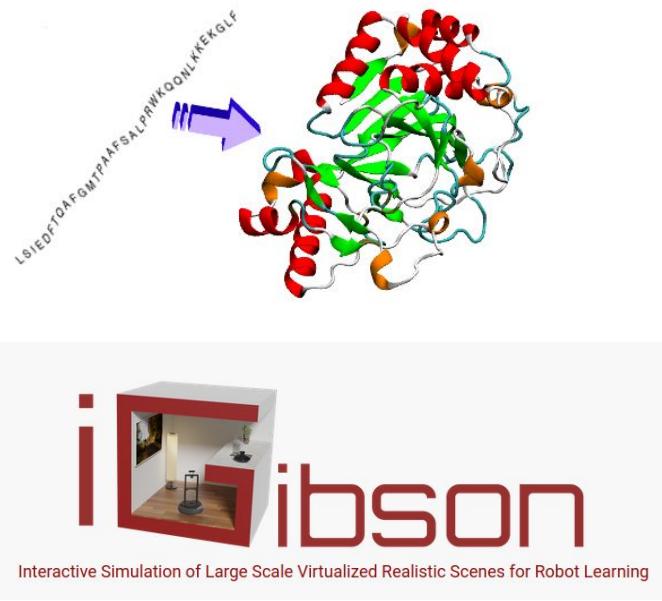
The Input



words



images



codes

actions

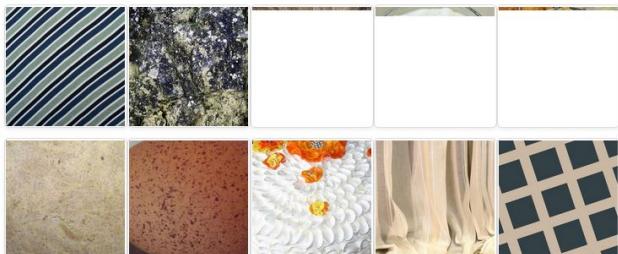
A screenshot of Microsoft Excel showing a spreadsheet titled "Business Expenses". The table includes columns for "Name", "Category", and several expense types like "Internet Services", "Cell Phone Service", "Landline", etc. The "Totals" row shows a total of \$2,886.42 for expenses and \$4,351.10 for income. The "Insuran" column is partially cut off at the end of the table.

numbers

Not All Datasets are created Equal!



Describable Textures Dataset (DTD)



ImageNet

Chairs



Chairs by rotation



Chairs by background



ObjectNet

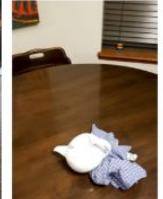
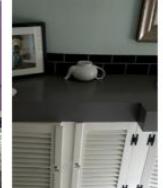
Chairs by viewpoint



Teapots



T-shirts



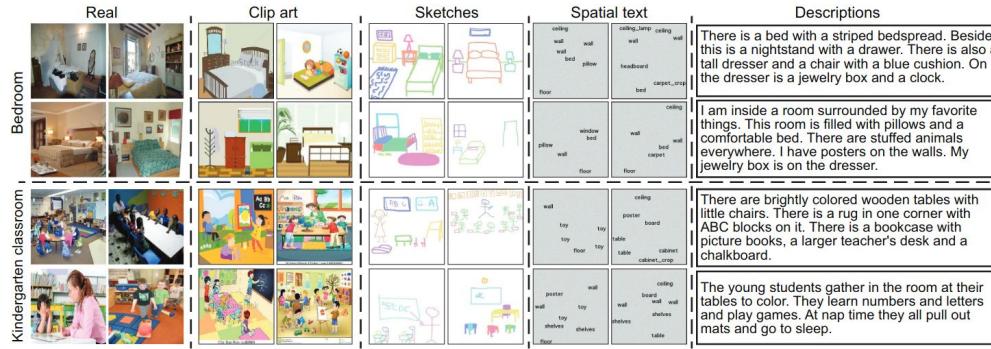


Figure 1: **Can you recognize scenes across different modalities?** Above, we show a few examples of our new cross-modal scene dataset. In this paper, we investigate how to learn cross-modal scene representations.

Castrejon, Aytar et al., NeurIPS 2016.

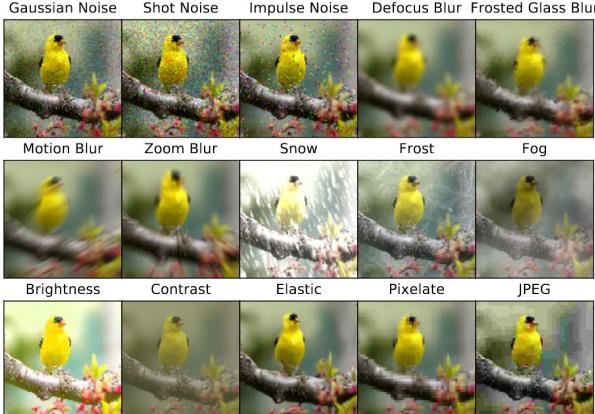
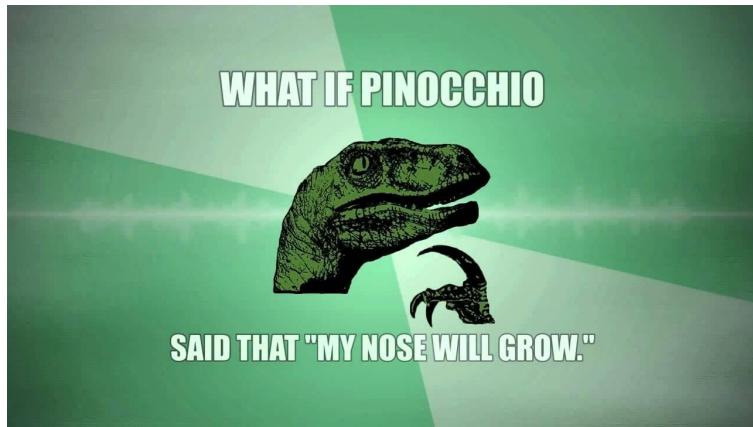


Figure 1: Our IMAGENET-C dataset consists of 15 types of algorithmically generated corruptions from noise, blur, weather, and digital categories. Each type of corruption has five levels of severity, resulting in 75 distinct corruptions. See different severity levels in Appendix B.

Hendrycks & Dietterich. ICLR 2019.



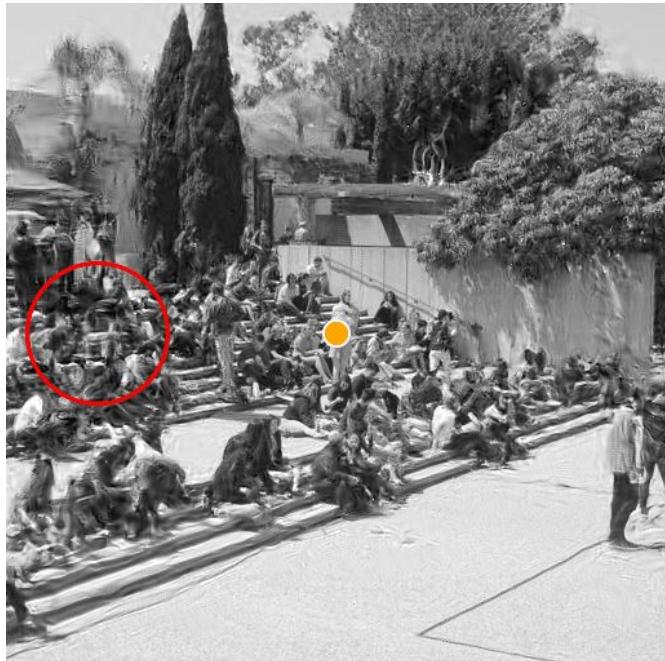
(a) Example viral images.



(b) Example non-viral images.

Deza & Parikh, CVPR 2015.

Case Study : What Dataset [*types of images*] should we use to study the effects of Foveated (Spatially-Adaptive) visual processing?



Case Study : What Dataset [*types of images*] should we use to study the effects of Foveated (Spatially-Adaptive) visual processing?



Finding Perceptual Divergences between Man + Machine: The Texture Bias

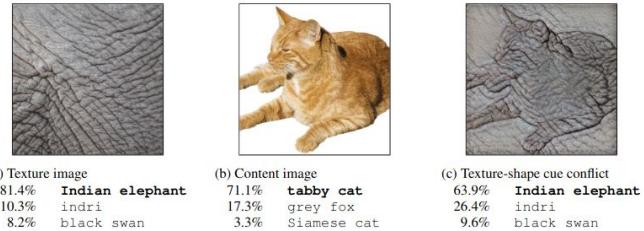


Figure 1: Classification of a standard ResNet-50 of (a) a texture image (elephant skin: only texture cues); (b) a normal image of a cat (with both shape and texture cues), and (c) an image with a texture-shape cue conflict, generated by style transfer between the first two images.



Robert Geirhos

Finding Perceptual Divergences between Man + Machine: The Texture Bias

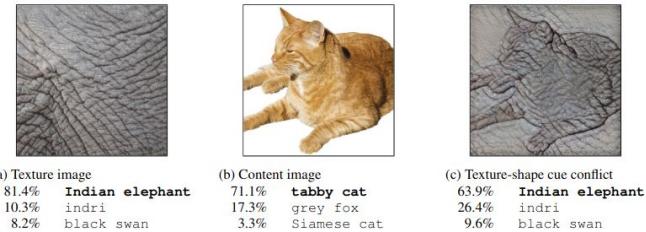
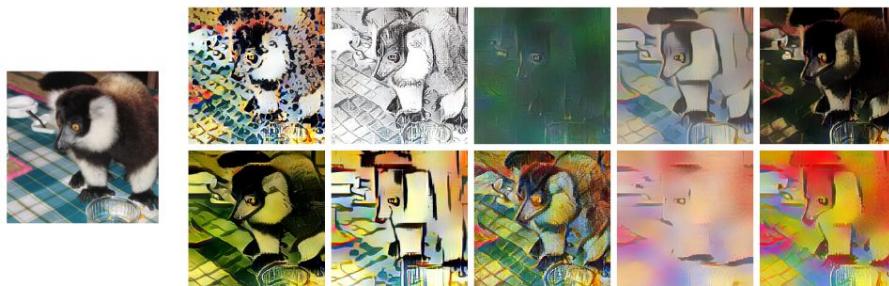


Figure 1: Classification of a standard ResNet-50 of (a) a texture image (elephant skin: only texture cues); (b) a normal image of a cat (with both shape and texture cues), and (c) an image with a texture-shape cue conflict, generated by style transfer between the first two images.



Robert Geirhos

Finding Perceptual Divergences between Man + Machine: The Texture Bias

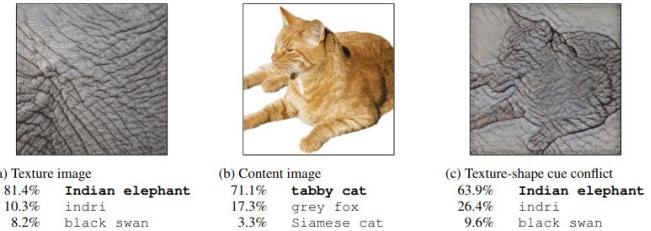


Figure 1: Classification of a standard ResNet-50 of (a) a texture image (elephant skin: only texture cues); (b) a normal image of a cat (with both shape and texture cues), and (c) an image with a texture-shape cue conflict, generated by style transfer between the first two images.

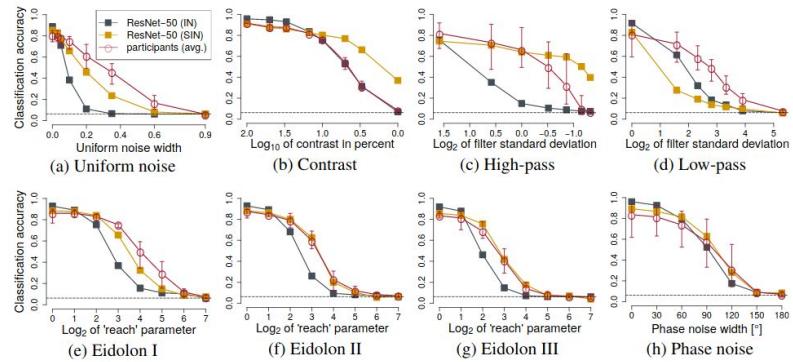


Figure 6: Classification accuracy on parametrically distorted images. ResNet-50 trained on Stylized-ImageNet (SIN) is more robust towards distortions than the same network trained on ImageNet (IN).



Finding Perceptual Divergences between Man + Machine: The Texture Bias



Figure 1: Classification of a standard ResNet-50 of (a) a texture image (elephant skin: only texture cues); (b) a normal image of a cat (with both shape and texture cues), and (c) an image with a texture-shape cue conflict, generated by style transfer between the first two images.

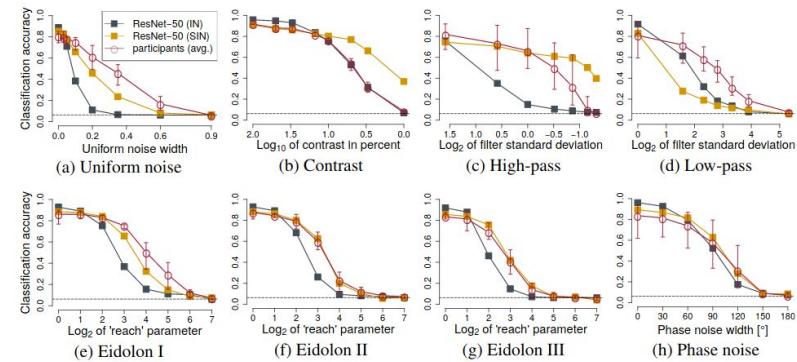
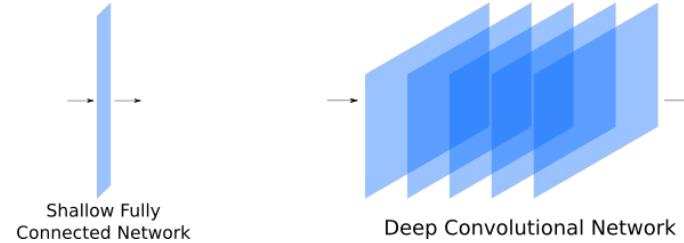


Figure 6: Classification accuracy on parametrically distorted images. ResNet-50 trained on Stylized-ImageNet (SIN) is more robust towards distortions than the same network trained on ImageNet (IN).

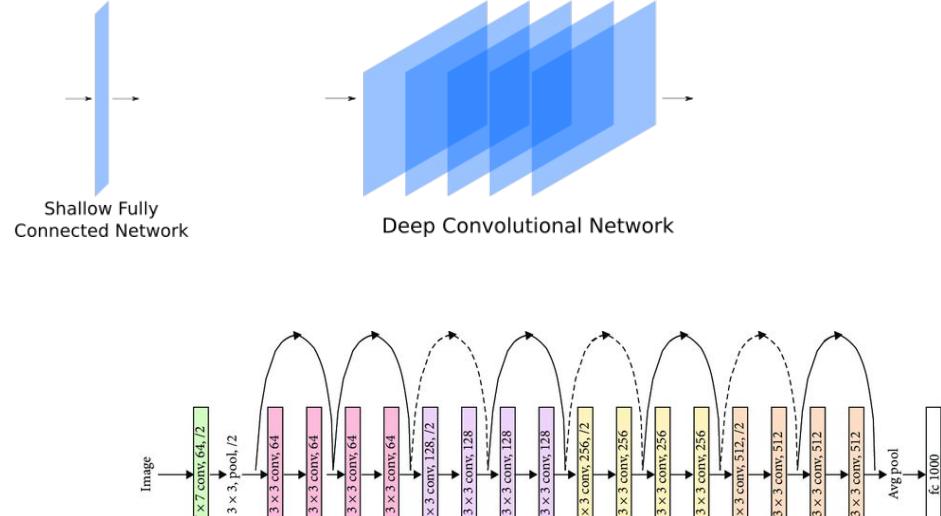
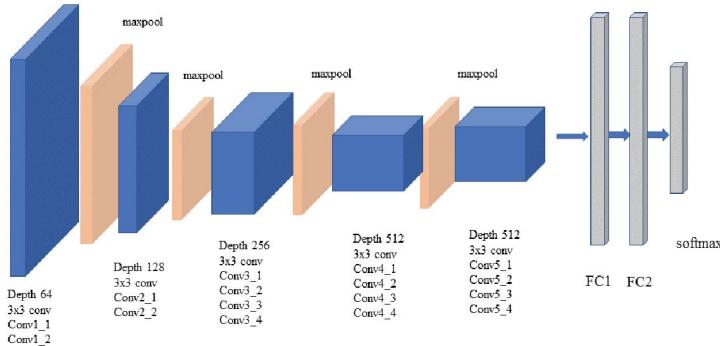


Robert Geirhos

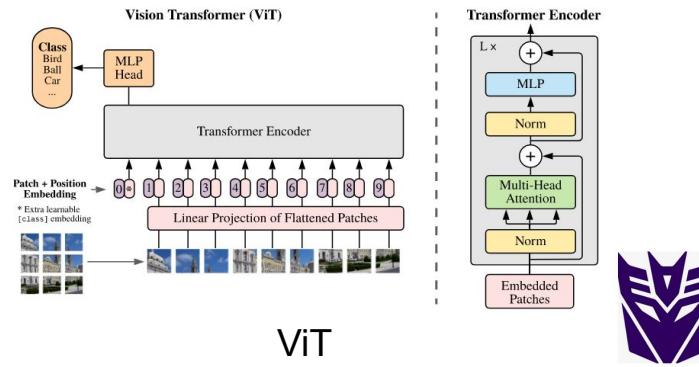
The Architecture



The Architecture



VGG11



Why is the Architecture important?

Why is the Architecture important?

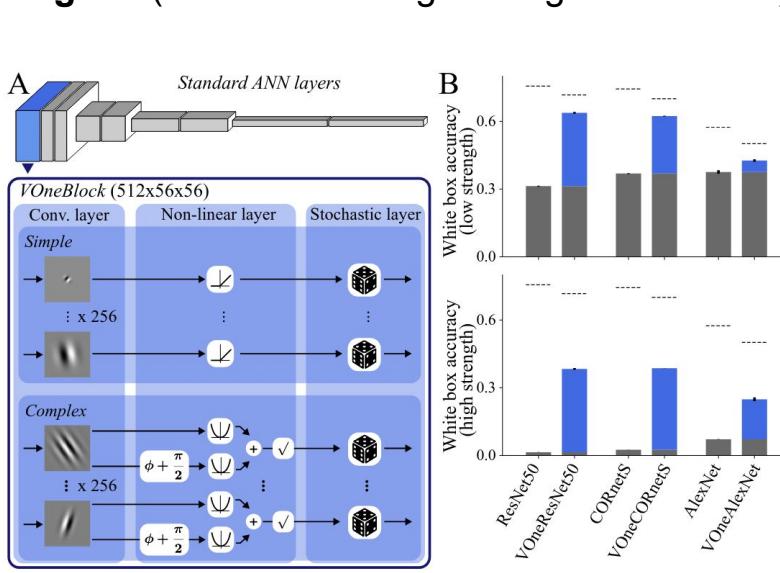
Different Architectures can enable better expressiveness and **approximation power**

Approximation by Superpositions of a Sigmoidal Function*

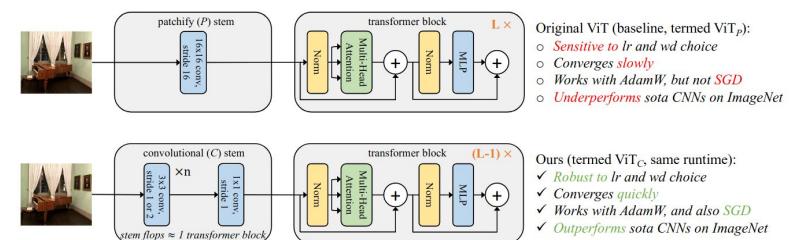
G. Cybenko†

Abstract. In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of n real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. Our results settle an open question about representability in the class of single hidden layer neural networks. In particular, we show that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single internal, hidden layer and any continuous sigmoidal nonlinearity. The paper discusses approximation properties of other possible types of nonlinearities that might be implemented by artificial neural networks.

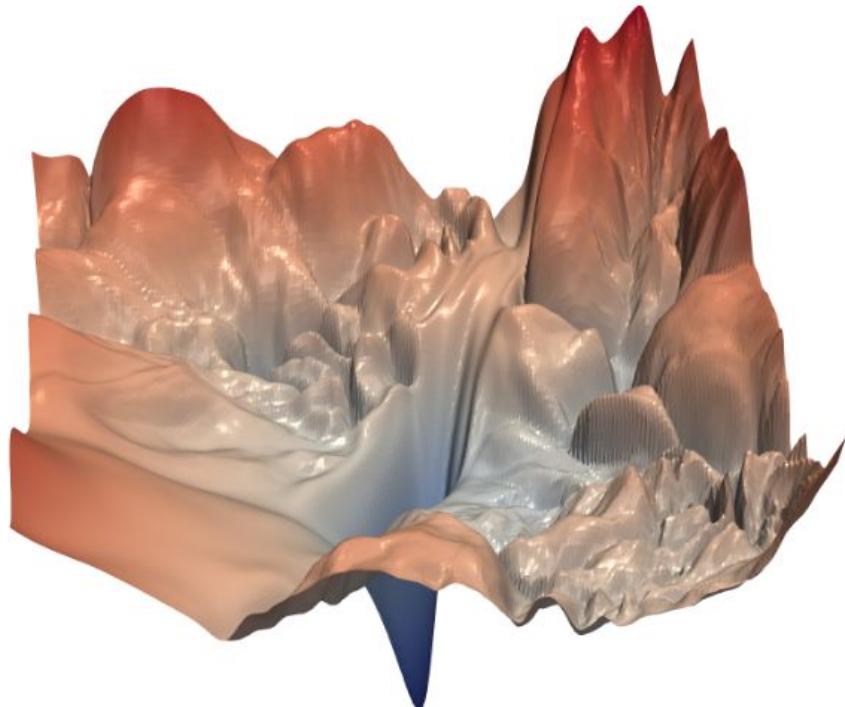
Key words. Neural networks, Approximation, Completeness.



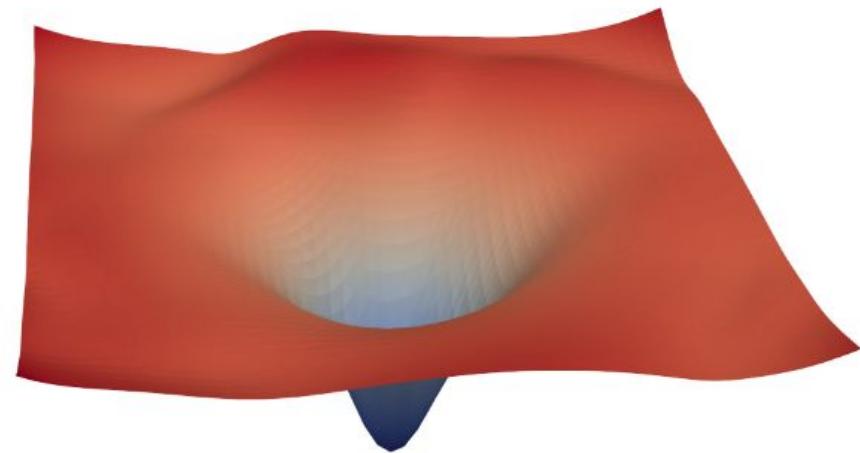
Some architectures could provide better **stability** in learning



Effect of Architecture on the Loss Landscape!



(a) without skip connections



(b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

The Representation

Is usually imposed via the latent space in the architecture

Can sometimes be imposed via a transform in a 2-stage model (a.k.a. Inductive bias)

But also goes back to other deep questions like: “How should we represent an image?” (or signal)

Should it be represented as a Matrix full of doubles from 0 to 1?

Should it be represented as a Matrix full of positives integers from 0 to 255?

What Color Space should be used? RGB? Lab? Why?

Should we represent images as vectors instead of Matrices, or Sets? Why?

Example 1: The case of Point Clouds

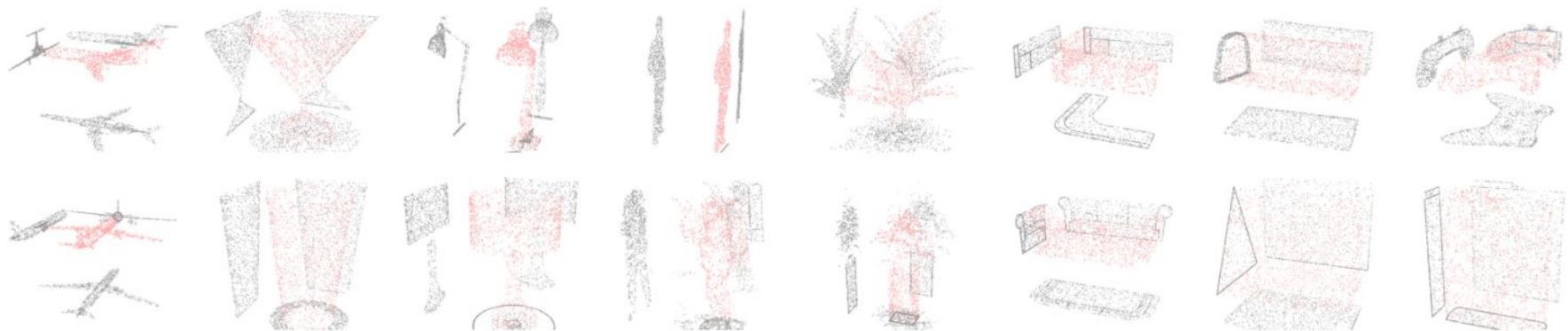


Figure 2: Examples for 8 out of 40 object classes (column) in the ModelNet40. Each point-cloud is produced by sampling 1000 particles from the mesh representation of the original MeodelNet40 instances. Two point-clouds in the same column are from the same class. The projection of particles into xy, zy and xz planes are added for better visualization.

Example 1b:

Property 1 A function $f : 2^{\mathfrak{X}} \rightarrow \mathcal{Y}$ acting on sets must be permutation **invariant** to the order of objects in the set, i.e. for any permutation $\pi : f(\{x_1, \dots, x_M\}) = f(\{x_{\pi(1)}, \dots, x_{\pi(M)}\})$.

In the supervised setting, given N examples of $X^{(1)}, \dots, X^{(N)}$ as well as their labels $y^{(1)}, \dots, y^{(N)}$, the task would be to classify/regress (with variable number of predictors) while being permutation invariant w.r.t. predictors. Under unsupervised setting, the task would be to assign high scores to valid sets and low scores to improbable sets. These scores can then be used for set expansion tasks, such as image tagging or audience expansion in field of computational advertisement. In *transductive* setting, each instance $x_m^{(n)}$ has an associated labeled $y_m^{(n)}$. Then, the objective would be instead to learn a permutation **equivariant** function $\mathbf{f} : \mathfrak{X}^M \rightarrow \mathcal{Y}^M$ that upon permutation of the input instances permutes the output labels, i.e. for any permutation π :

$$\mathbf{f}([x_{\pi(1)}, \dots, x_{\pi(M)}]) = [f_{\pi(1)}(\mathbf{x}), \dots, f_{\pi(M)}(\mathbf{x})] \quad (1)$$

Example 2:

Perceptual Systems

A. Input Foveation Transform



$$f_*(\circ)$$



CNN-backbone
Scene Classification

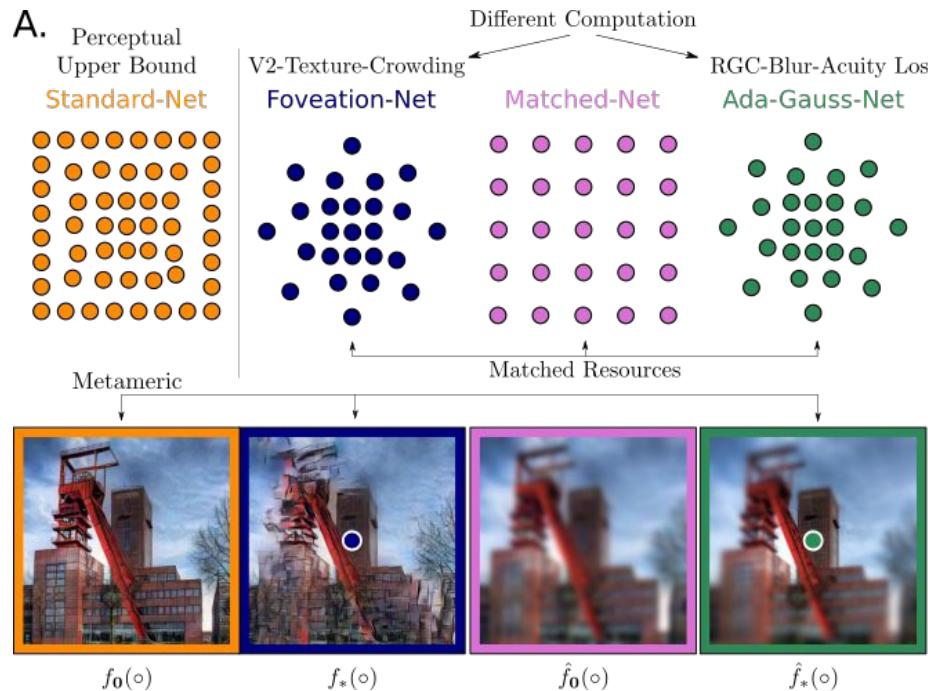


$$f_0(\circ)$$



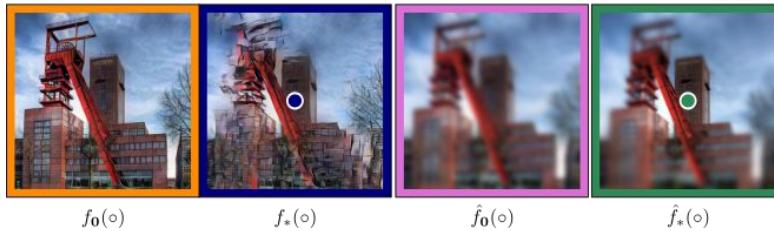
We perceptually evaluate each network!

The Representation will affect Generalization capacity!

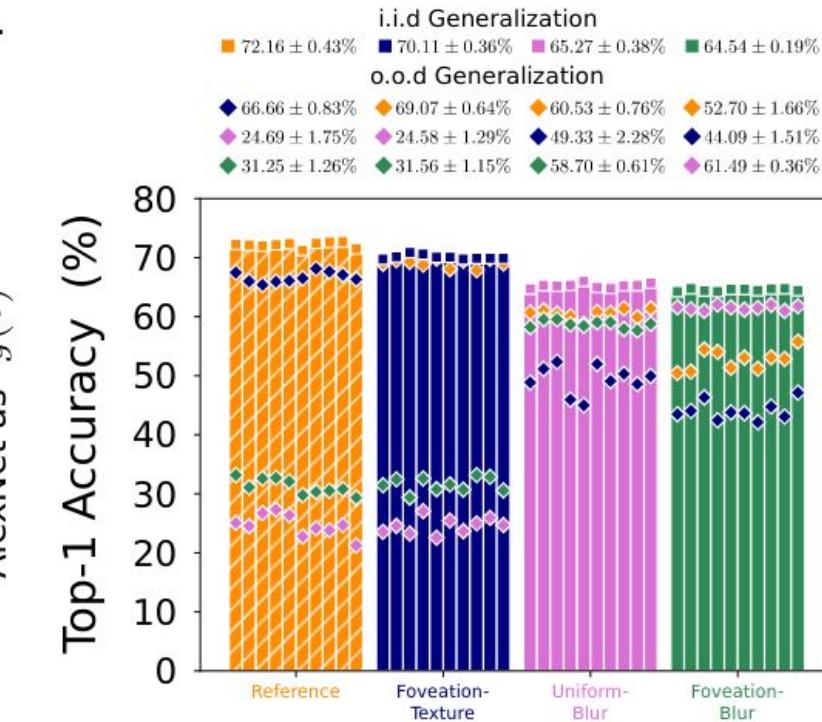


Talia
Konkle

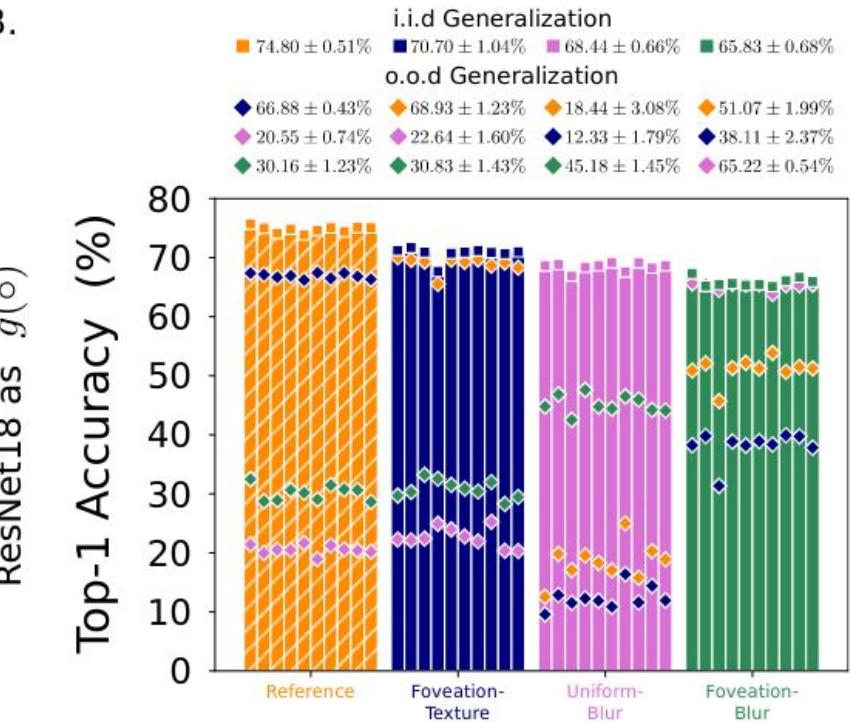




A.



B.



The Output (also known as the Task)

What can you compute/predict/estimate from this image?

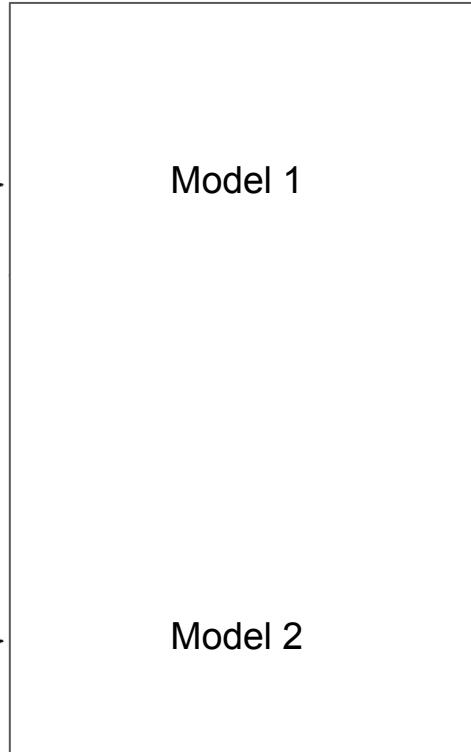


Fixing the Input ; Varying the Output

A.



Recognize Object
(Classification)



→ Frog

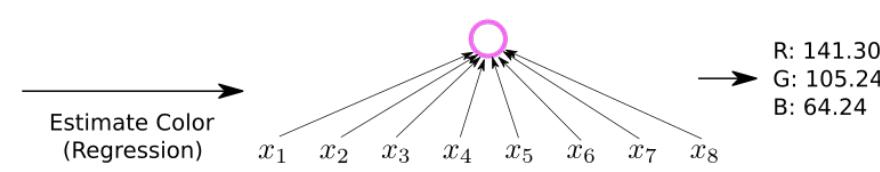
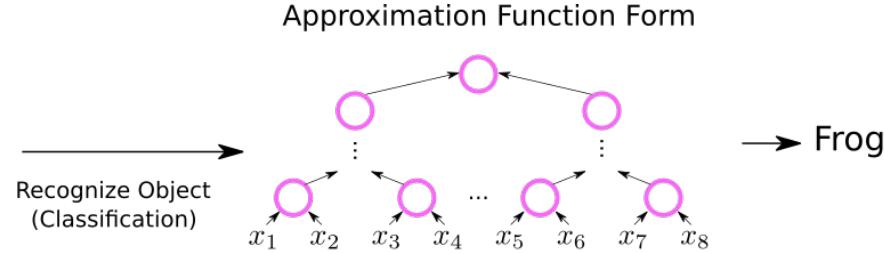


Estimate Color
(Regression)

→ R: 141.30
G: 105.24
B: 64.24

Fixing the Input ; Varying the Output

A.



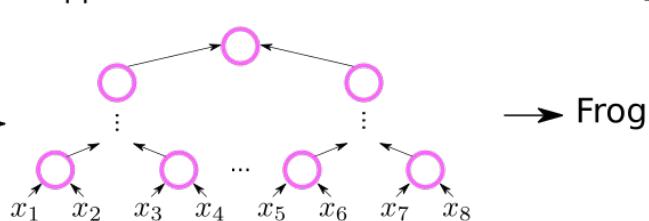
(And tying back to the role of Architecture)

A.



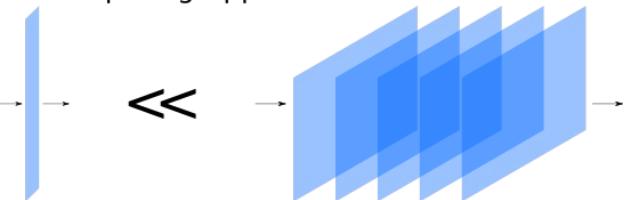
Recognize Object
(Classification)

Approximation Function Form

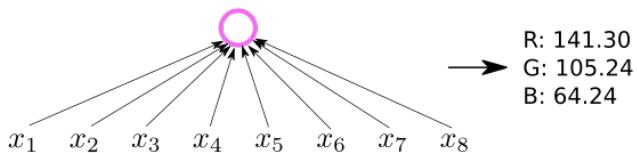


B.

Comparing Approximation Power



Estimate Color
(Regression)



Shallow Fully
Connected Network

>

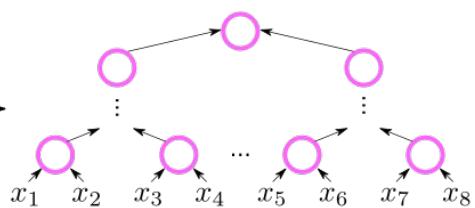
Deep Convolutional Network

A.



Recognize Object
(Classification)

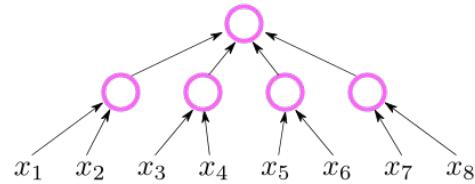
Approximation Function Form



→ Frog



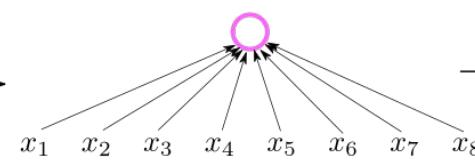
Recognize Texture
(Classification)



→ Frog



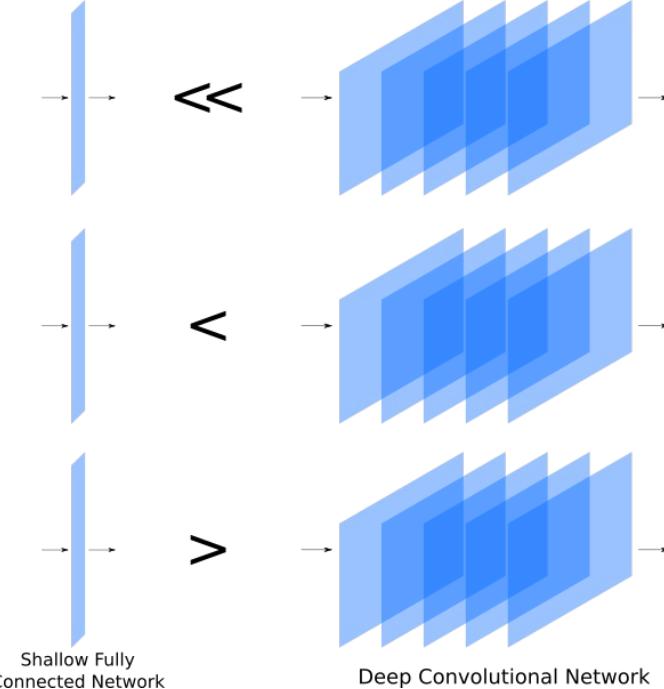
Estimate Color
(Regression)



R: 141.30
G: 105.24
B: 64.24

B.

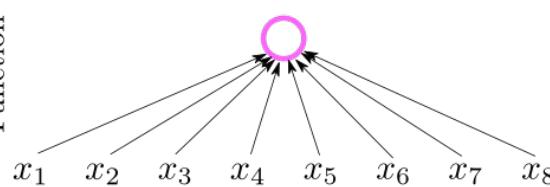
Comparing Approximation Power



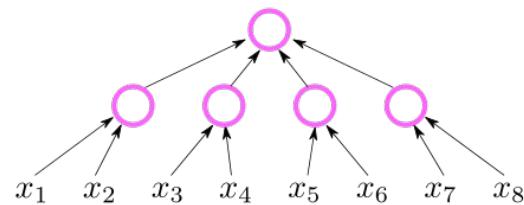
Expressiveness of Architecture should match Task!

Task

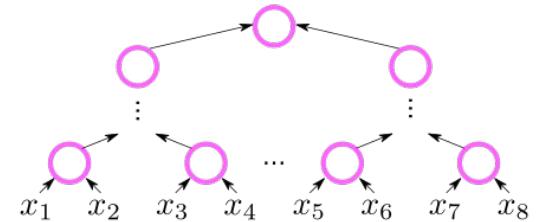
Global & Non-Compositional



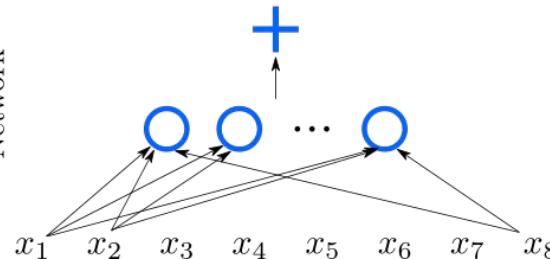
Local & Compositional



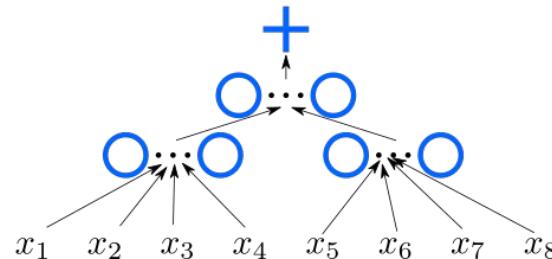
Local & Hierarchically Compositional



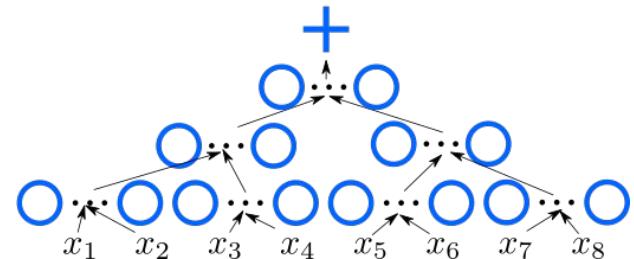
Shallow Network



Convolutional Network

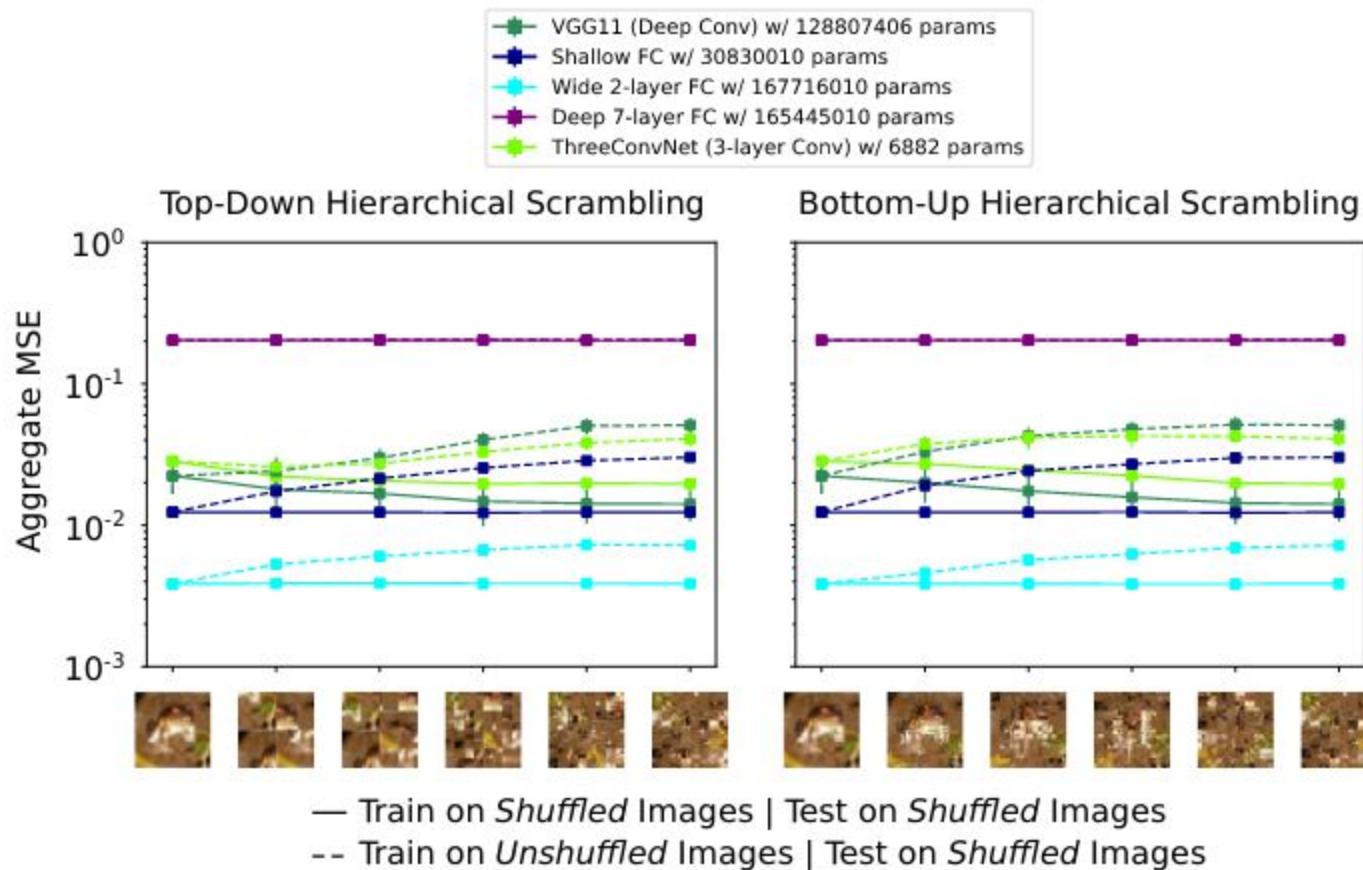


Deep Convolutional Network

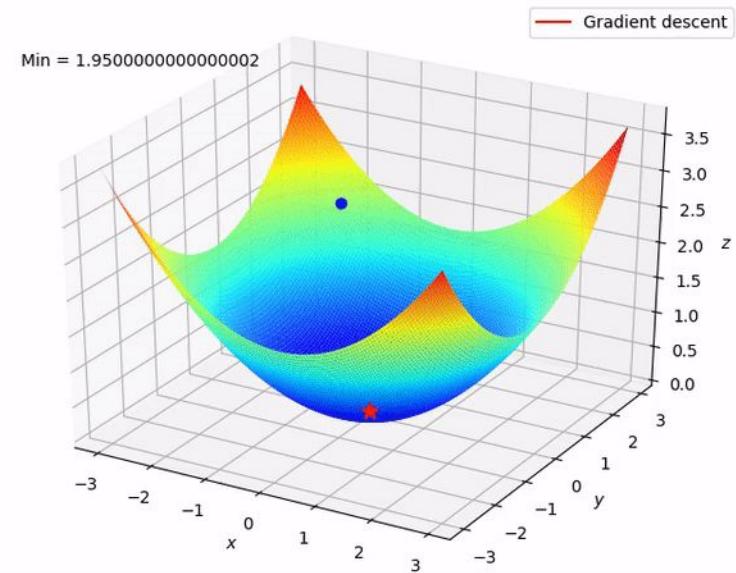
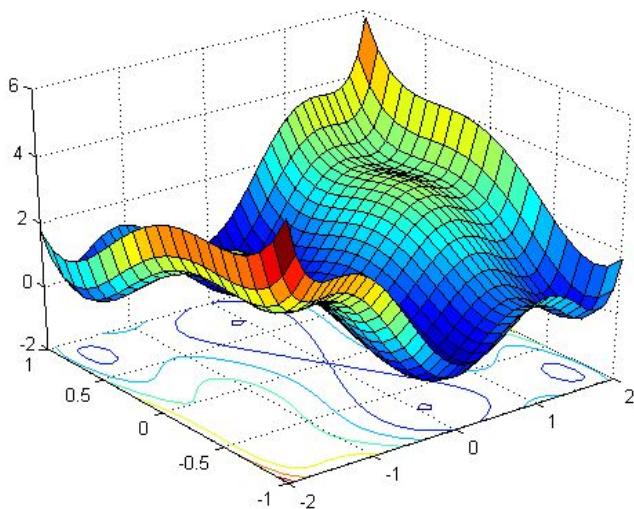


Network

Effects of Hierarchical Scrambling on Color Estimation



The Loss Surface (and function)



The Loss Surface (and function)

$$\text{Loss} = -\frac{1}{\frac{\text{output size}}{\text{size}}} \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

The Loss Surface (and function)

Cross Entropy Loss (Generally used in Classification)

$$\text{Loss} = -\frac{1}{\text{output size}} \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

↓
This is the simpler version!

$$\text{Loss} = - \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

The Loss Surface (and function)

Cross Entropy Loss (Generally used in Classification)

$$\text{Loss} = -\frac{1}{\text{output size}} \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

↓
This is the simpler version!

$$\text{Loss} = - \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

White-Board Example: GT: [0,0,1]

S1: [0.05,0.05,0.9] ; S2: [0.4,0.3,0.3]

The Loss Surface (and function)

Mean Square Error (Generally used in Regression*)

$$\text{Loss} = \sum_{i=1}^{\text{output size}} (y_i - \hat{y}_i)^2$$

White-Board Example: GT: [0,0,1]

S1: [0.05,0.05,0.9] ; S2: [0.4,0.3,0.3]

*but can also be used for Classification ;)

Note: Each Loss Function penalizes error **differently**

White-Board Example: GT: [0,0,1]

S1: [0.05,0.05,0.9] ; S2: [0.4,0.3,0.3]

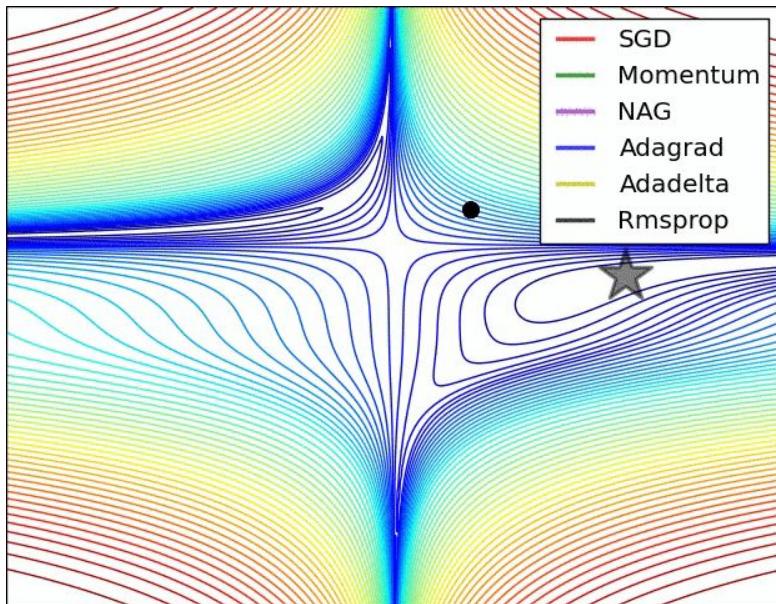
$$\text{CE: } \text{Loss} = - \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

$$\text{MSE: } \text{Loss} = \sum_{i=1}^{\text{output size}} (y_i - \hat{y}_i)^2$$

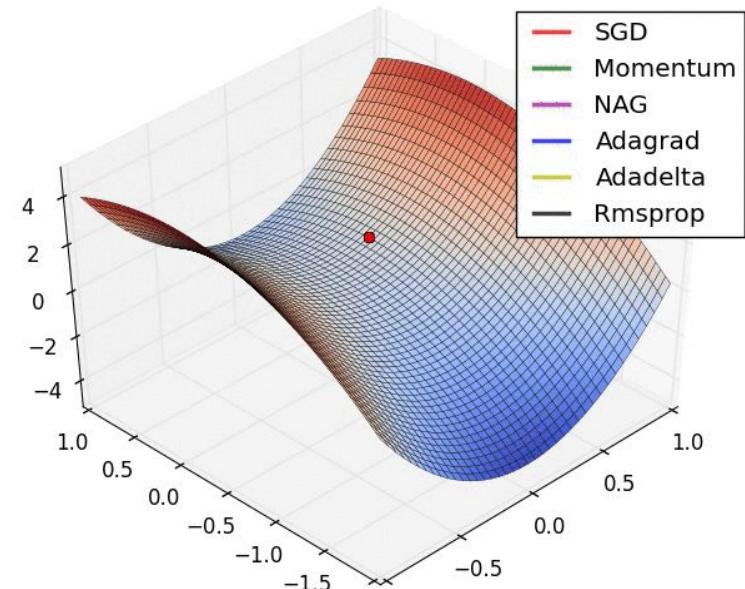
*but can also be used for Classification ;)

The Optimization procedure (Learning Rule)

Optimizers on Surface Contours

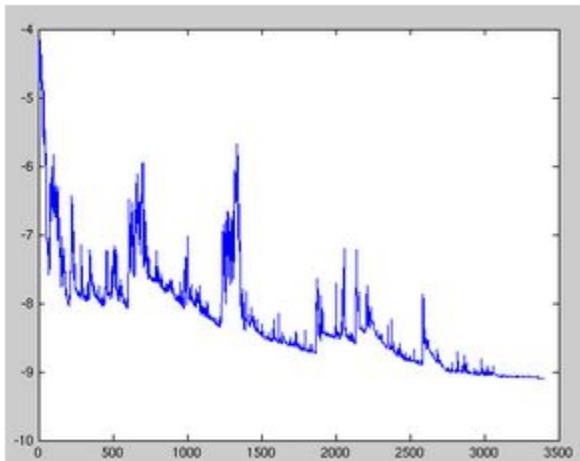


Optimizers on Saddle Point



Stochastic Gradient Descent:

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x^{(i)}; y^{(i)}).$$

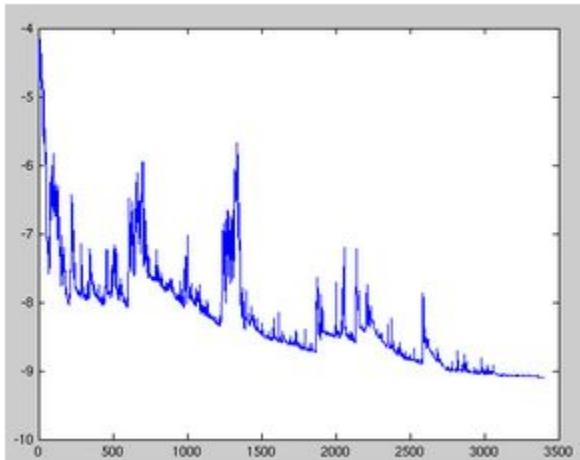


The Optimization Zoo!

- Gradient descent optimization algorithms
 - Momentum
 - Nesterov accelerated gradient
 - Adagrad
 - Adadelta
 - RMSprop
 - Adam
 - AdaMax
 - Nadam
 - AMSGrad
- Other recent optimizers
- Visualization of algorithms
- Which optimizer to use?

Stochastic Gradient Descent:

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x^{(i)}; y^{(i)}).$$



Healthy Sign of Training: **Loss goes Down!**

Good News: Optimizers in PyTorch are 1 line of code!

The Optimization Zoo!

- Gradient descent optimization algorithms
 - Momentum
 - Nesterov accelerated gradient
 - Adagrad
 - Adadelta
 - RMSprop
 - Adam
 - AdaMax
 - Nadam
 - AMSGrad
- Other recent optimizers
- Visualization of algorithms
- Which optimizer to use?



Next themes to be covered
during this week ... [!]

Why do we use Optimization and link it to Learning?

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton†
& Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California,
San Diego, La Jolla, California 92093, USA

† Department of Computer Science, Carnegie-Mellon University,
Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure¹.

There have been many attempts to design self-organizing neural networks. The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain. The task is specified by giving the desired state vector of the output units for each state vector of the input units. If the input units are directly connected to the output units it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections so as to progressively reduce the difference between the actual and desired output vectors². Learning becomes more interesting but

more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input-output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

The total input, x_j , to unit j is a linear function of the outputs, y_i , of the units that are connected to j and of the weights, w_{ji} , on these connections

$$x_j = \sum_i y_i w_{ji} \quad (1)$$

Units can be given biases by introducing an extra input to each unit which always has a value of 1. The weight on this extra input is called the bias and is equivalent to a threshold of the opposite sign. It can be treated just like the other weights.

A unit has a real-valued output, y_j , which is a non-linear function of its total input

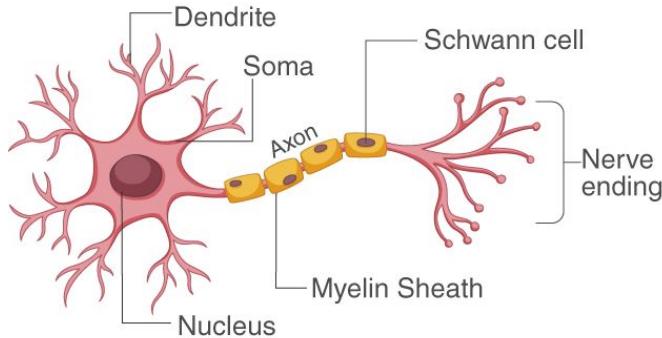
$$y_j = \frac{1}{1 + e^{-x_j}} \quad (2)$$

† To whom correspondence should be addressed.

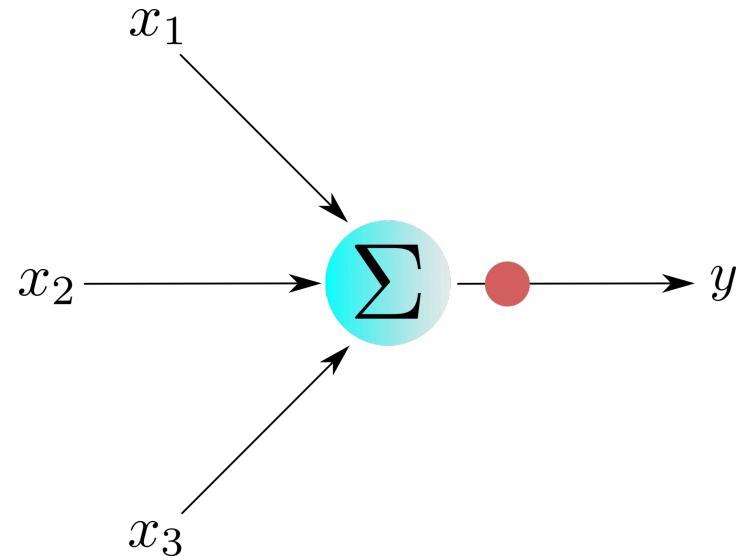
The Neuron : It's basic structure

Structure of a Biological Neuron

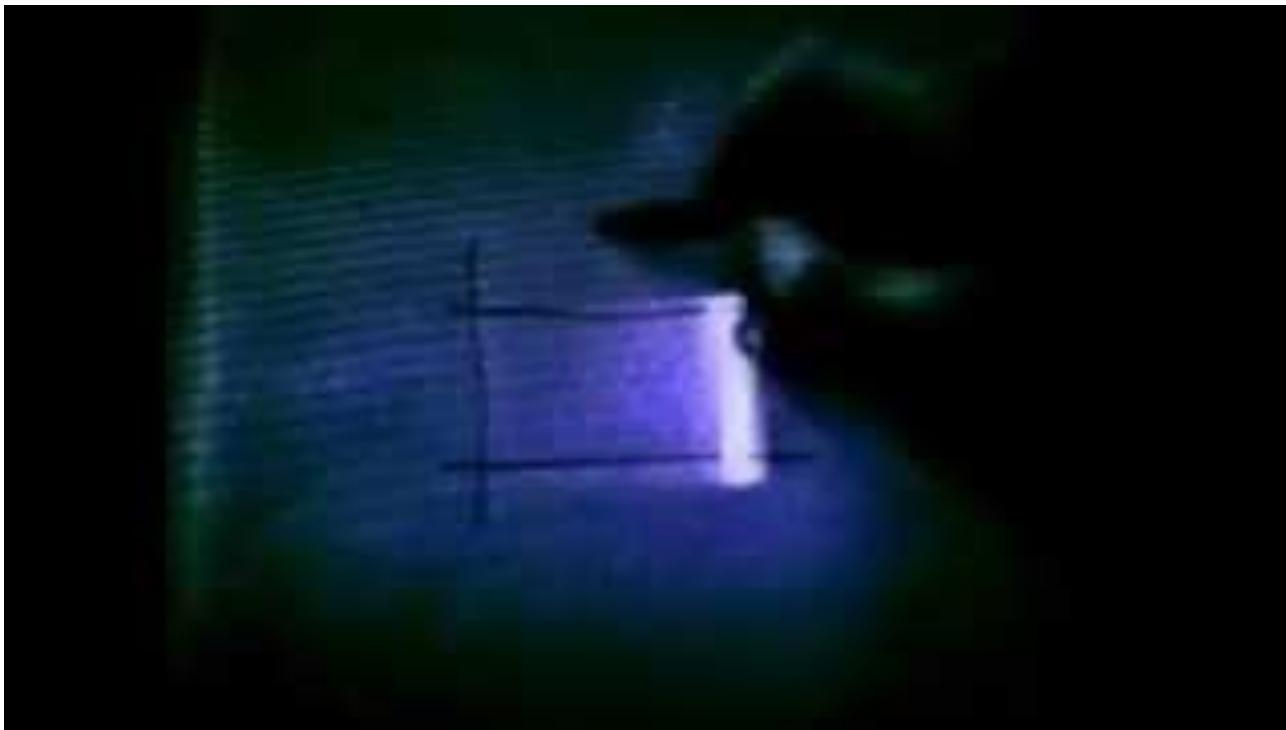
STRUCTURE OF NEURON



Structure of an Artificial Neuron



Hubel & Wiesel (Nobel Prize). Experiments 1982.



Hubel & Wiesel (Nobel Prize). Experiments 1982.



What is the equivalent firing rate in a Machine? (In a Deep Neural Network in particular?)

The Three Levels of Marr

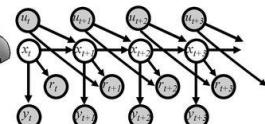
Three levels of description (*David Marr, 1982*)

Computational

Why do things work the way they do?
What is the goal of the computation?
What are the unifying principles?

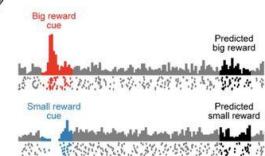
maximize:

$$R_t = r_{t+1} + r_{t+2} + \dots + r_T$$



Algorithmic

What representations can implement such computations?
How does the choice of representations determine the algorithm?



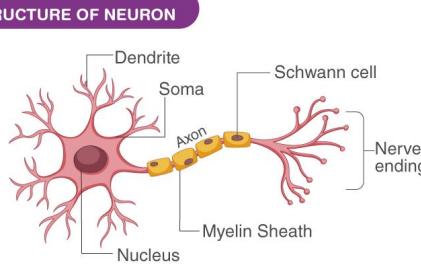
Implementational

How can such a system be built in hardware?
How can neurons carry out the computations?

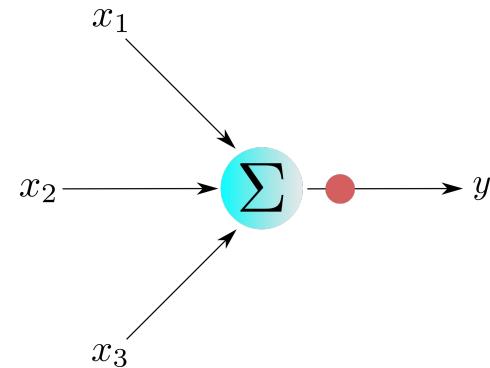


Structure of a Neuron

Structure of a Biological Neuron



Structure of an Artificial Neuron



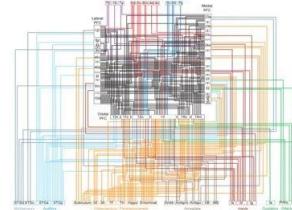
Homework (Will come in the Quiz) : Marr's 3 levels

Another Example / Otro ejemplo?

Computational
Theory



Representation and
Algorithm



Hardware
Implementation

