Deep Learning - Vision <-> NLP

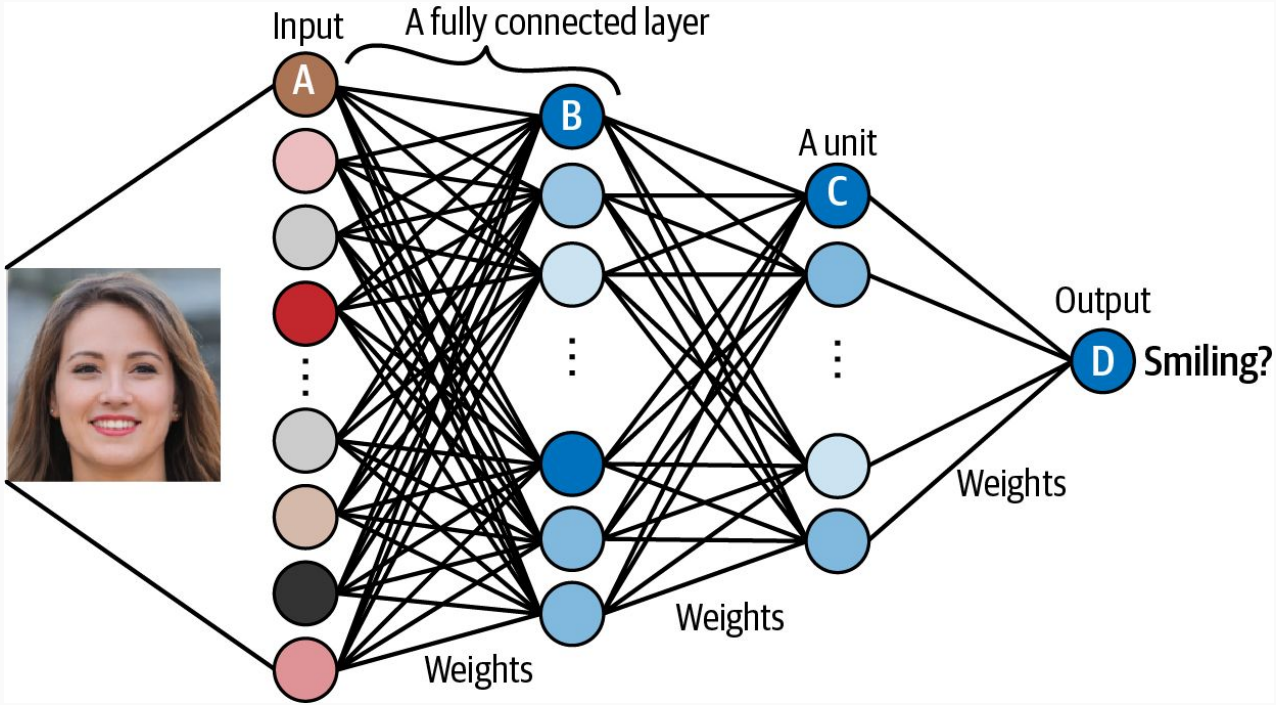
Vijay Raghavan

Agenda

1. Deep Learning in Vision
    a. Recap of CNNs.
    b. Foundational Models in Vision
    c. Key Learnings from Vision
2. Convergence of Deep Learning in NLP and Vision
3. Semi-supervised Methods for Data - Contrastive Learning
4. Future

Vision

Smiling?



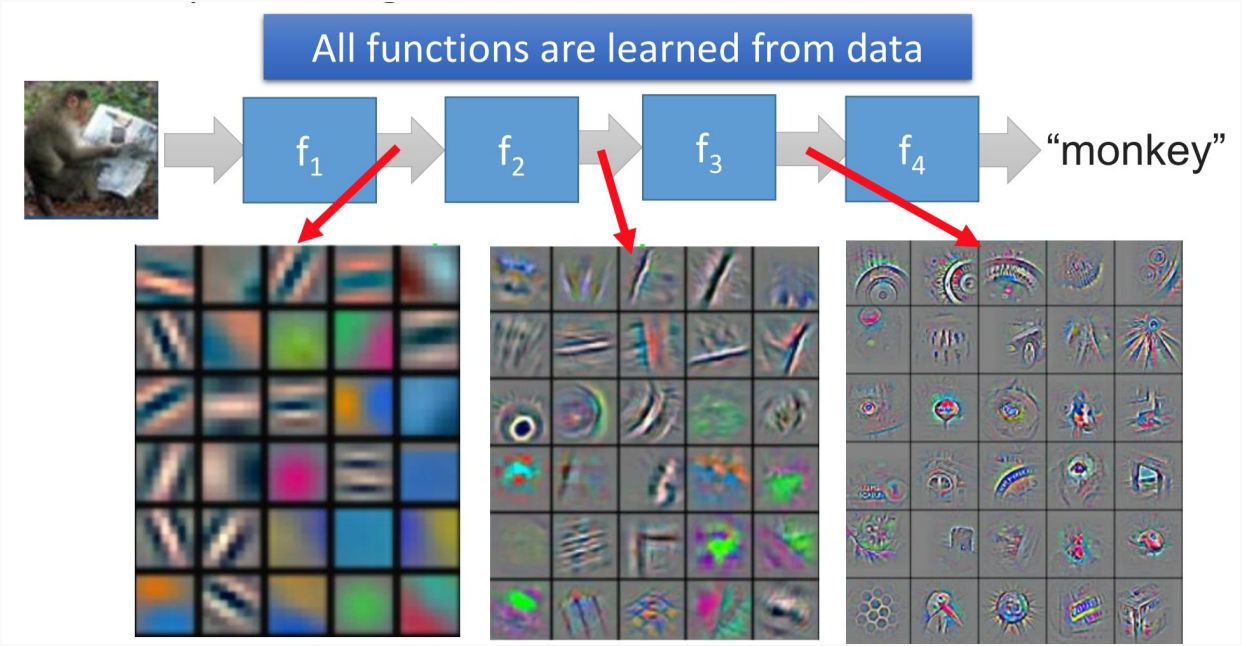
Foster, D. (2022). *Generative deep learning*. " O'Reilly Media, Inc.".

Learning High-Level Features

- Unit A receives the value for an individual channel of an input pixel.
- Unit B combines its input values so that it fires strongest when a particular low-level feature such as an edge is present.
- Unit C combines the low-level features so that it fires strongest when a higher-level feature such as teeth are seen in the image.
- Unit D combines the high-level features so that it fires strongest when the person in the original image is smiling.

Foster, D. (2022). *Generative deep learning*. " O'Reilly Media, Inc.".

Learning

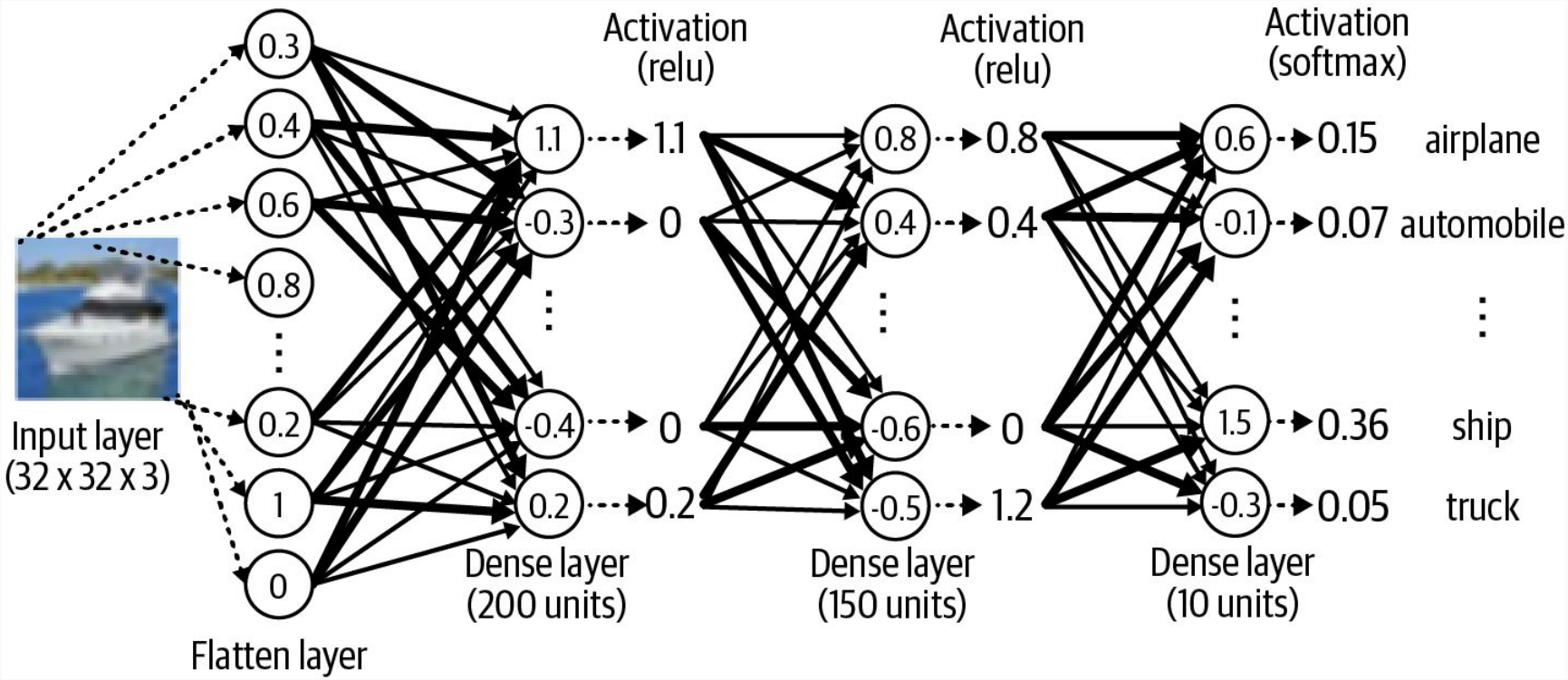


[Source] Courtesy by Prof. HY Lee (NTU)

CIFAR 10


Airplane
Automobile
Bird
Cat
Deer
Dog
Frog
Horse
Ship
Truck

A Vanilla Neural Network



Foster, D. (2022). *Generative deep learning*. " O'Reilly Media, Inc.".

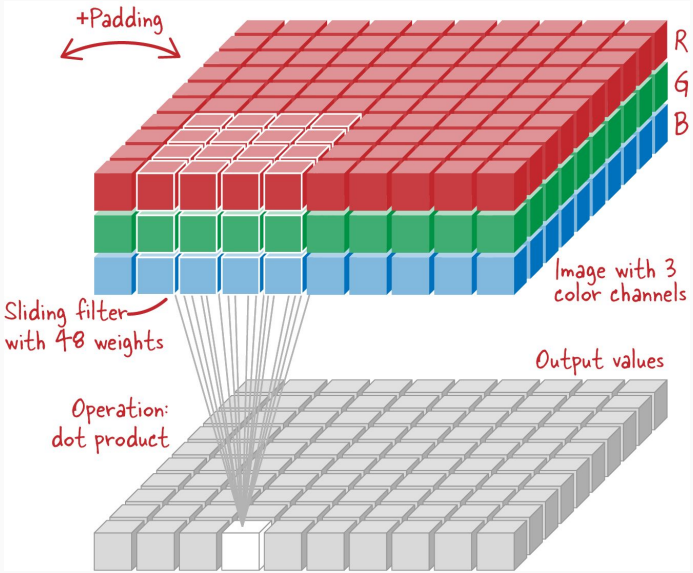
CIFAR 10 - Vanilla Neural Network

Foster, D. (2022). *Generative deep learning*. " O'Reilly Media, Inc.".

Convolutional Neural Network

What is a Convolution?

- Convolutional layers were designed specifically for images.
- They operate in two dimensions and can capture shape information; they work by sliding a small window, called a convolutional filter, across the image in both directions.
- A convolution is an mathematical operation between a small matrix (kernel) and a larger matrix (input)
- The kernel is slid across the input matrix
- At each spatial location, the dot product is computed between the kernel and a patch of the input
- This dot product result forms the output activation map
- Convolutions enable efficient localization and extraction of patterns

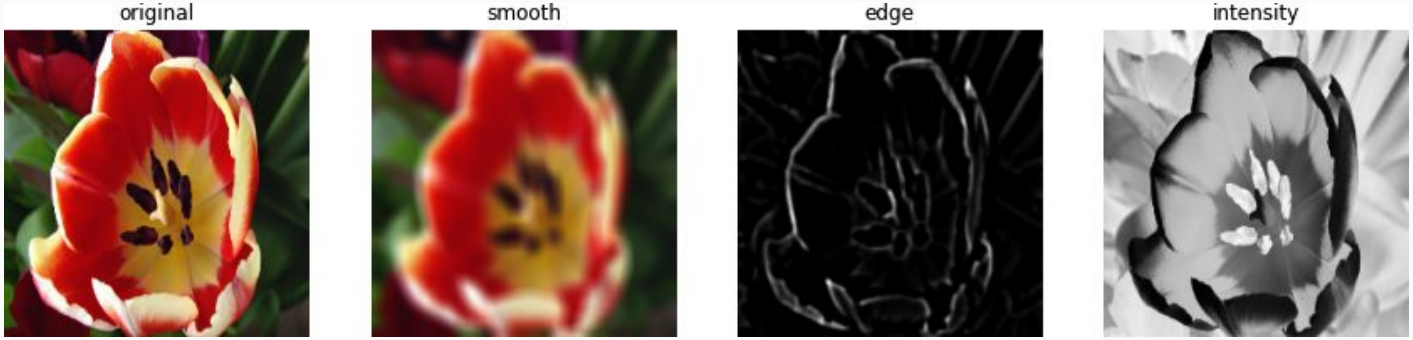


Processing an image with a single 4x4 convolutional filter. The filter slides across the image in both directions, producing one output value at each position.

Lakshmanan, V., Görner, M., & Gillard, R. (2021). *Practical machine learning for computer vision*. " O'Reilly Media, Inc."

Convolutional filters can achieve many different effects

- A filter where all the weights are the same is a “smoothing” filter - each pixel within a window has an equal contribution for the resulting output pixel.
- Other filters can use correlations and anti-correlations of adjacent pixels to compute new information about the image to identify textures and edges.

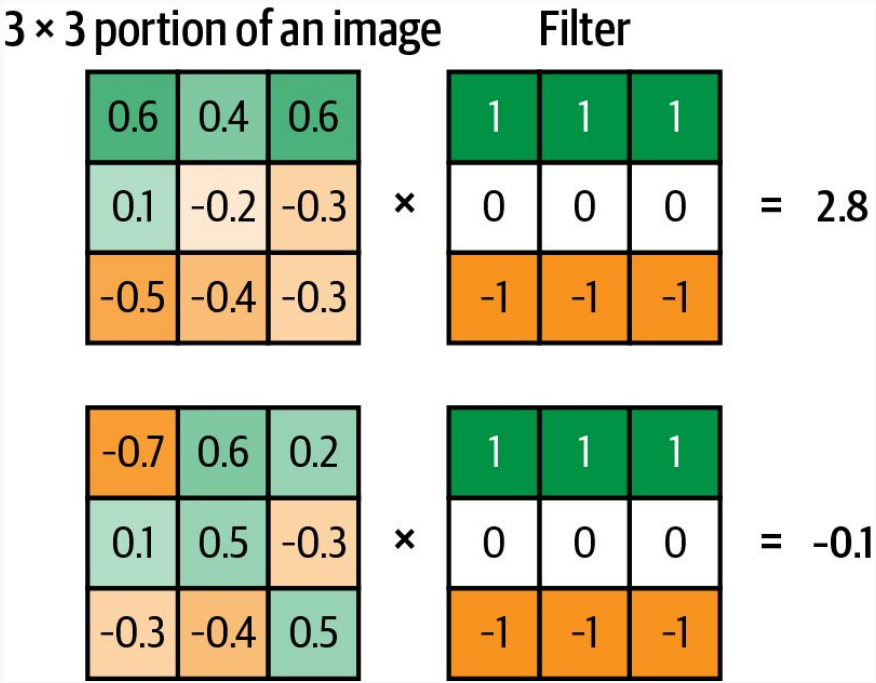


Lakshmanan, V., Görner, M., & Gillard, R. (2021). *Practical machine learning for computer vision*. " O'Reilly Media, Inc."

Convolutional layers

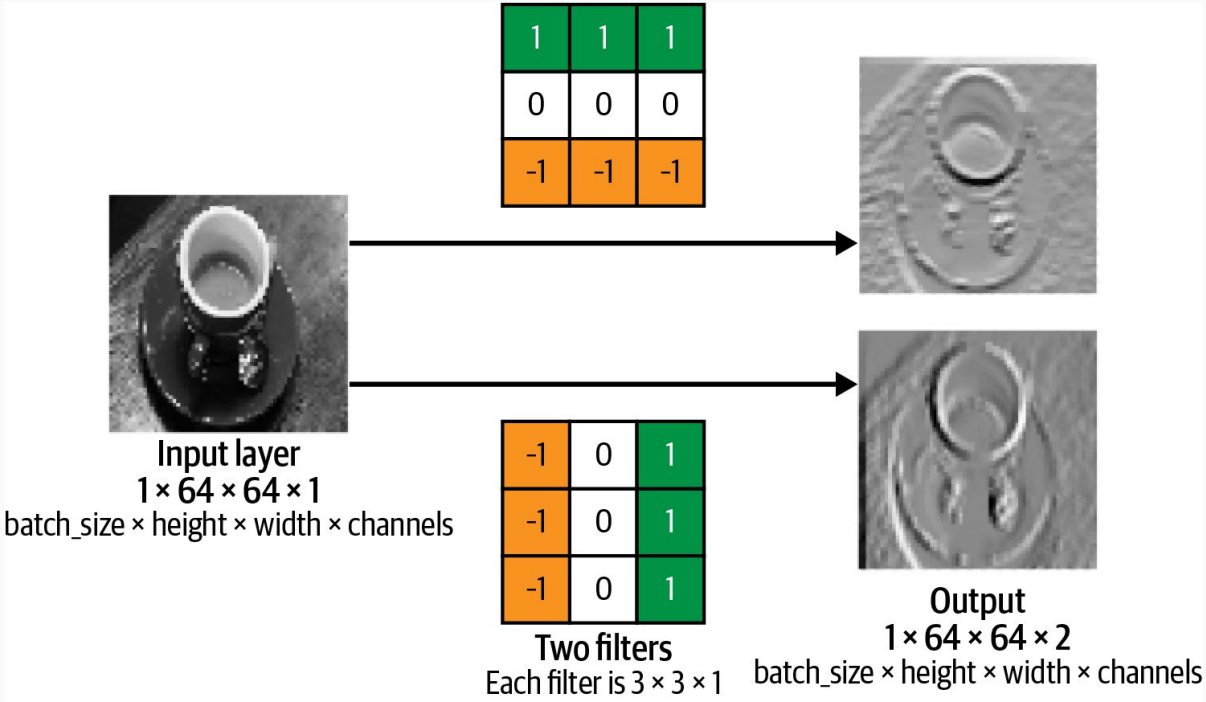
- A single convolutional filter can process an entire image with very few learnable parameters
- It will not be able to learn and represent enough of the complexities of the image and multiple filters are needed.
- A convolutional layer typically contains tens or hundreds of similar filters, each with its own independent learnable weights
- They are applied to the image in succession, and each produces a channel of output values.
- The output of a convolutional layer is a multichannel set of 2D values.

3 × 3 convolutional filter applied to 2 portions of a grayscale image



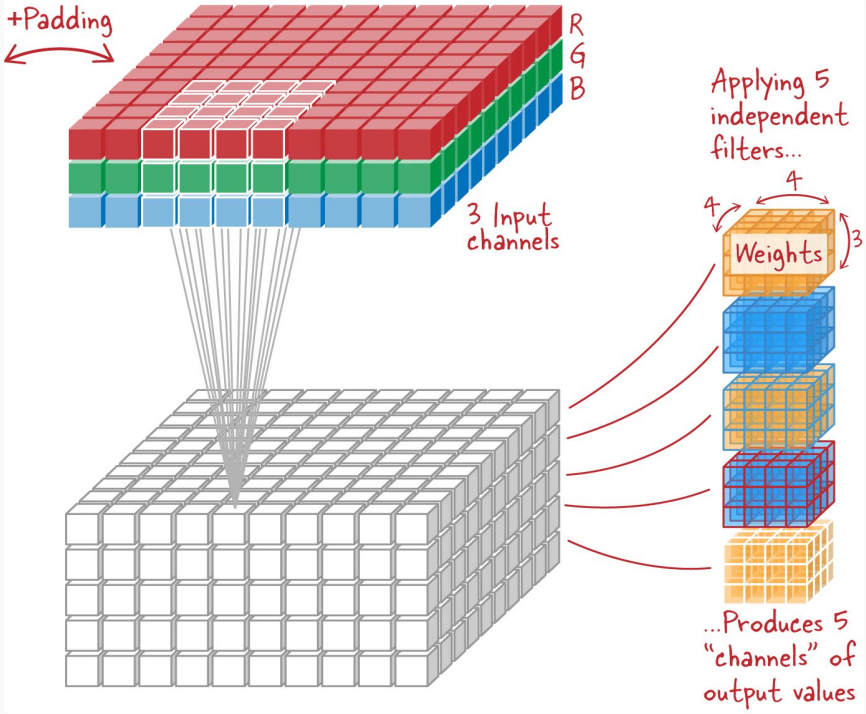
Foster, D. (2022). *Generative deep learning*. " O'Reilly Media, Inc.".

Two convolutional filters applied to a grayscale image

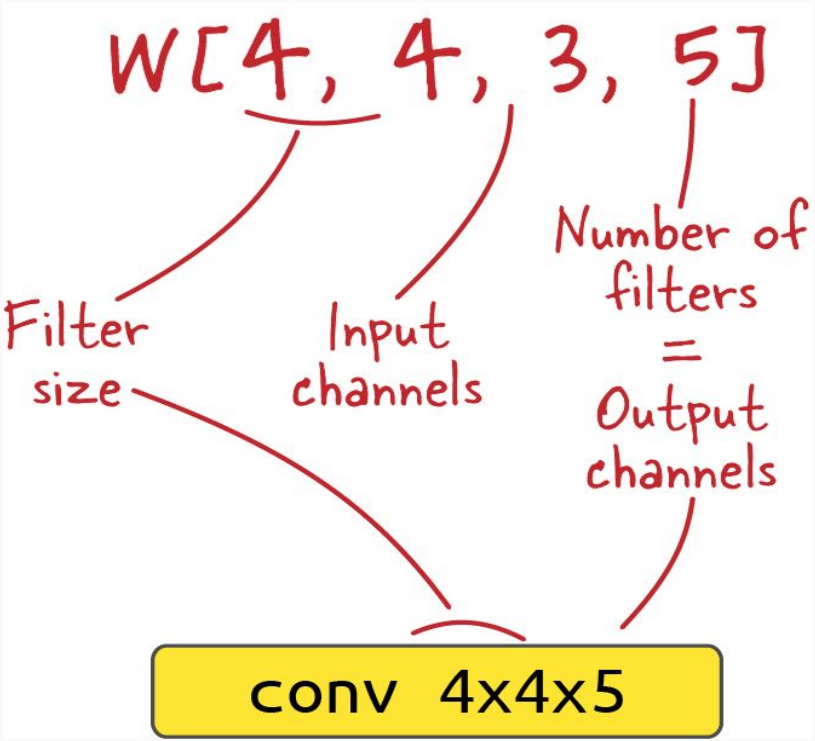


Foster, D. (2022). *Generative deep learning*. " O'Reilly Media, Inc.".

Processing an image with multiple convolutional filters



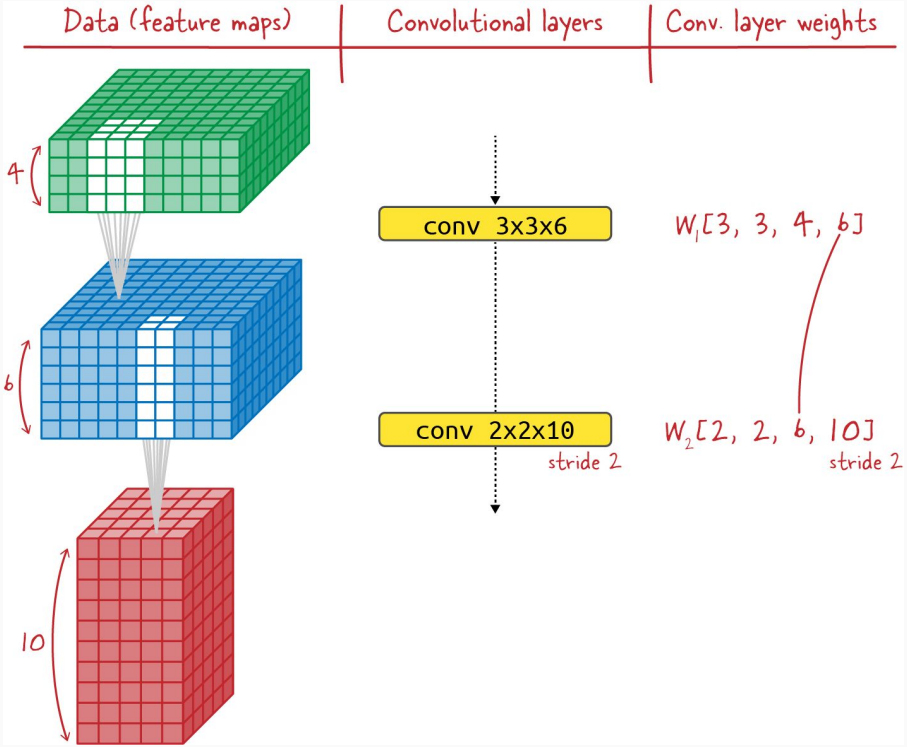
Filters are of the same size (here, 4x4x3) but have independent learnable weights



In this case, with 5 filters applied, the total number of learnable weights in this convolutional layer is 4 * 4 * 3 * 5 = 240.

Lakshmanan, V., Görner, M., & Gillard, R. (2021). *Practical machine learning for computer vision*. " O'Reilly Media, Inc."

Stacking the Convolutional Layers



- Data transformed by two convolutional layers applied in sequence.
- Learnable weights are shown on the right.
- The second convolutional layer is applied with a stride of 2 and has six input channels, matching the six output channels of the previous layer.

Lakshmanan, V., Görner, M., & Gillard, R. (2021). *Practical machine learning for computer vision*. " O'Reilly Media, Inc."

Parameters

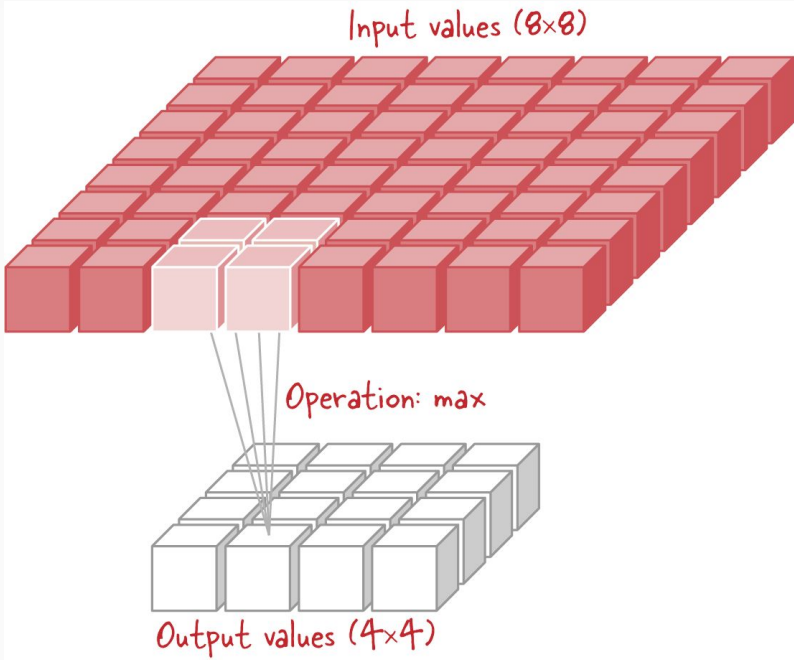
- filters - The number of independent filters to apply to the input.
- kernel_size - The size of each filter.
- strides - The filter slides across the input image in steps.
- padding - 'valid' for no padding or 'same' for zero-padding at the edges.
- activation - Like any neural network layer, a convolutional layer can be followed by an activation (nonlinearity).

```
tf.keras.layers.Conv2D(filters,
                       kernel_size,
                       strides=(1, 1),
                       padding='valid',
                       activation=None)
```

```
tf.keras.layers.Conv2D(filters=5,
kernel_size=4, padding='same')
```

Pooling Layers

- The goal of a neural network is usually to distill information
- Need layers to combine or downsample the information in each channel.
- The most commonly used downsampling operation is 2x2 max pooling.
- Max pooling - only the maximum value is retained for each group
- Average pooling - averages the four values



The max is taken for every group of 2x2 input values and the operation is repeated every two values in each direction (stride 2).

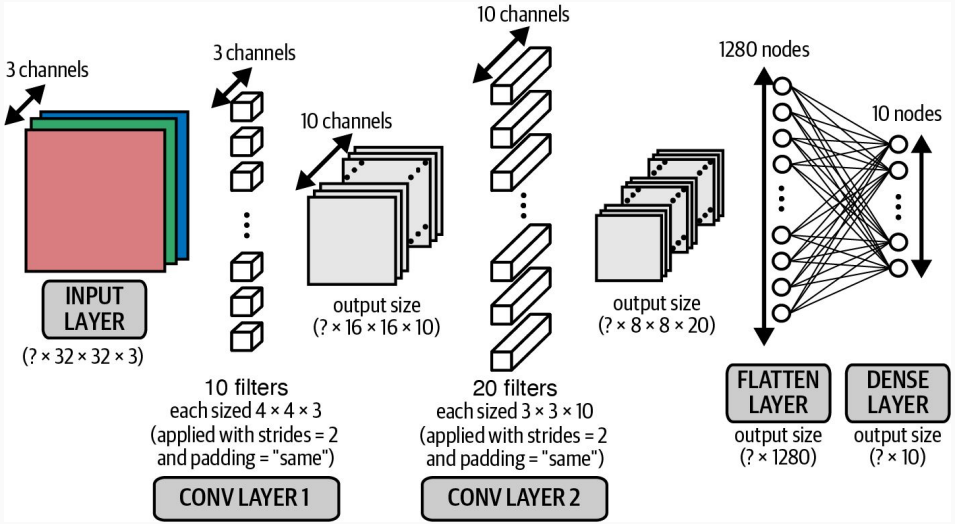
Lakshmanan, V., Görner, M., & Gillard, R. (2021). *Practical machine learning for computer vision*. " O'Reilly Media, Inc."

CIFAR 10 - CNN

Model Accuracy ~ 70%

Foster, D. (2022). *Generative deep learning*. " O'Reilly Media, Inc.".

Calculating Learnable Parameters



- First convolutional layer: (4 * 4 * 3 + 1) * 10 = 490 parameters
- Second convolutional layer: (3 * 3 * 10 + 1) * 20 = 1,820 parameters
- Flatten layer: No parameters
- Dense layer: 1,280 inputs * 10 outputs = 12,810 parameters

The total number of learnable parameters in this CNN model is:

490 + 1,820 + 0 + 12,810 = 15,120

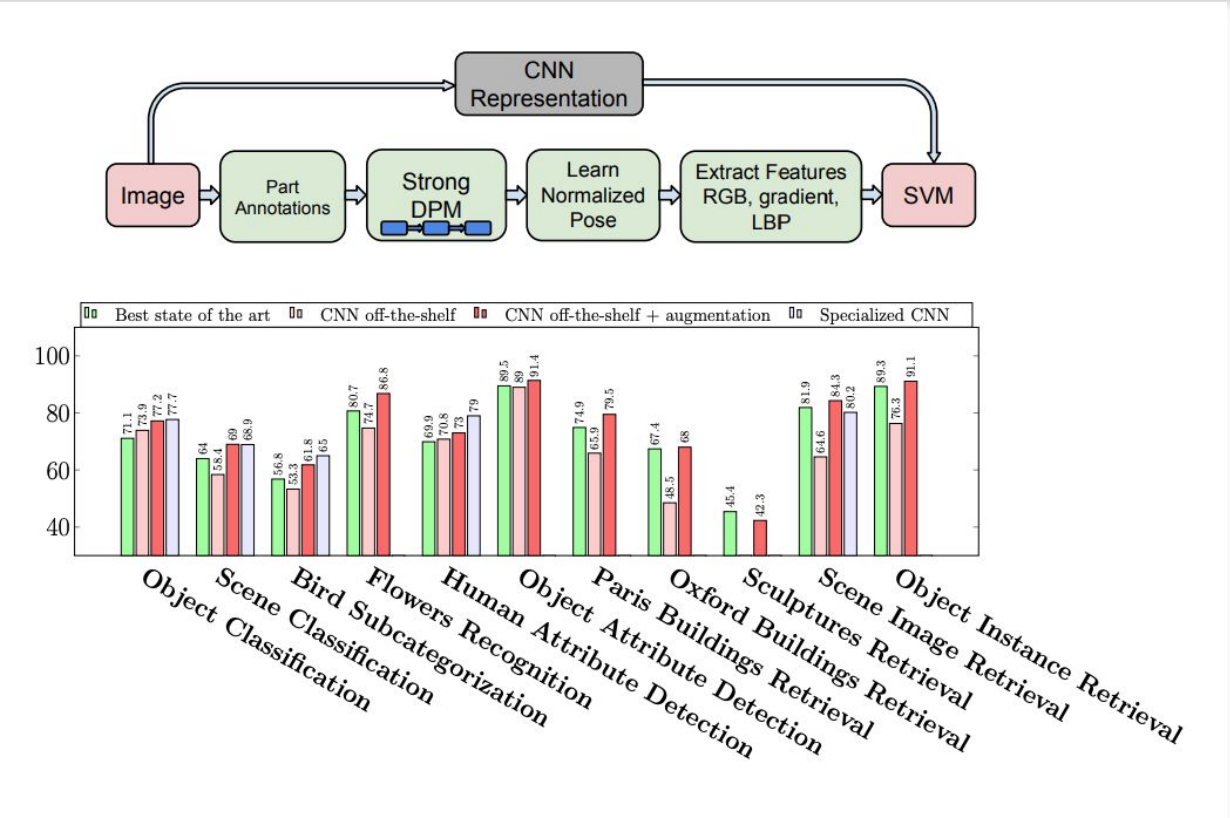
Foster, D. (2022). *Generative deep learning*. " O'Reilly Media, Inc.".

Key Takeaways

- For convolutional layers, the parameters come from the kernel/filter dimensions (height x width x input channels) plus a bias for each filter
- For dense/fully-connected layers, it's input units x output units
- Other layers like flatten, maxpool etc have no parameters

By going through each layer and calculating, we can find the total parameters which in this case is 15,120.

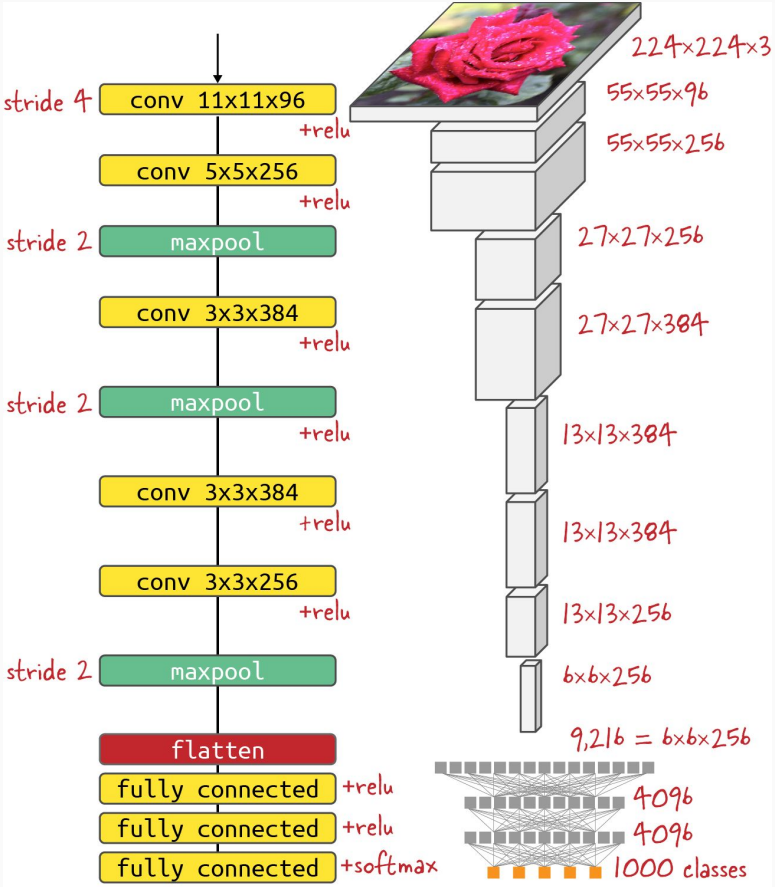
What's the big deal?



Sharif Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 806-813).

Foundational Models in Vision

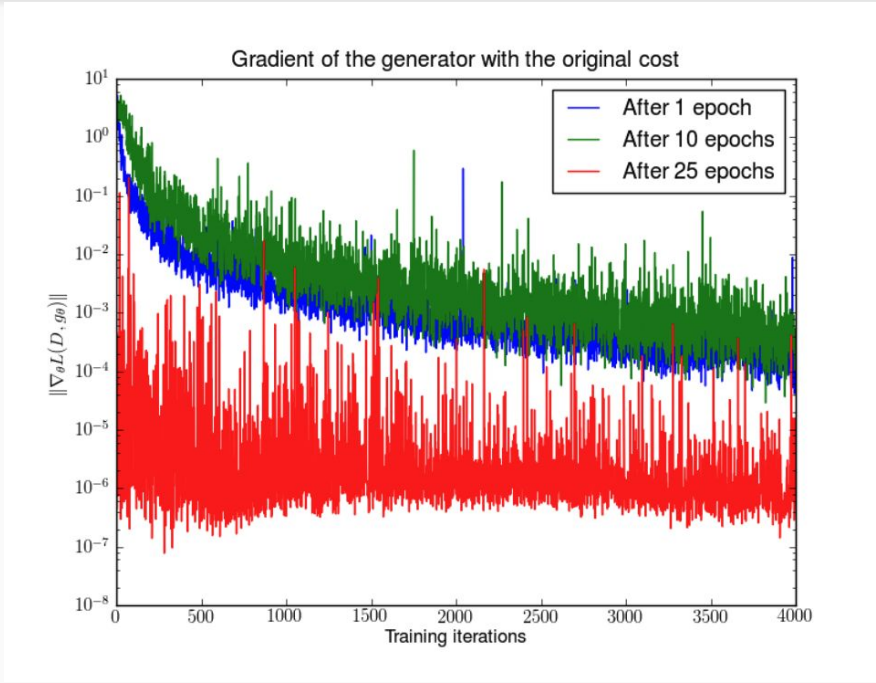
Alexnet



- Convolutional layers change the depth of the data—i.e., the number of channels.
- Max-pooling layers downsample the data in the height and width directions.
- AlexNet uses 3x3 max-pooling operations with a stride of 2.
- Every convolutional layer is activated by a ReLU activation function.
- The final four layers form the classification head of AlexNet, taking the last feature map, flattening all of its values into a vector, and feeding it through three fully connected layers.
- All convolutional and fully connected layers use an additive bias.
- When the ReLU activation function is used, it is customary to initialize the bias to a small positive value before training so that, after activation, all layers start with a nonzero output and a nonzero gradient

A common challenge with very deep neural networks

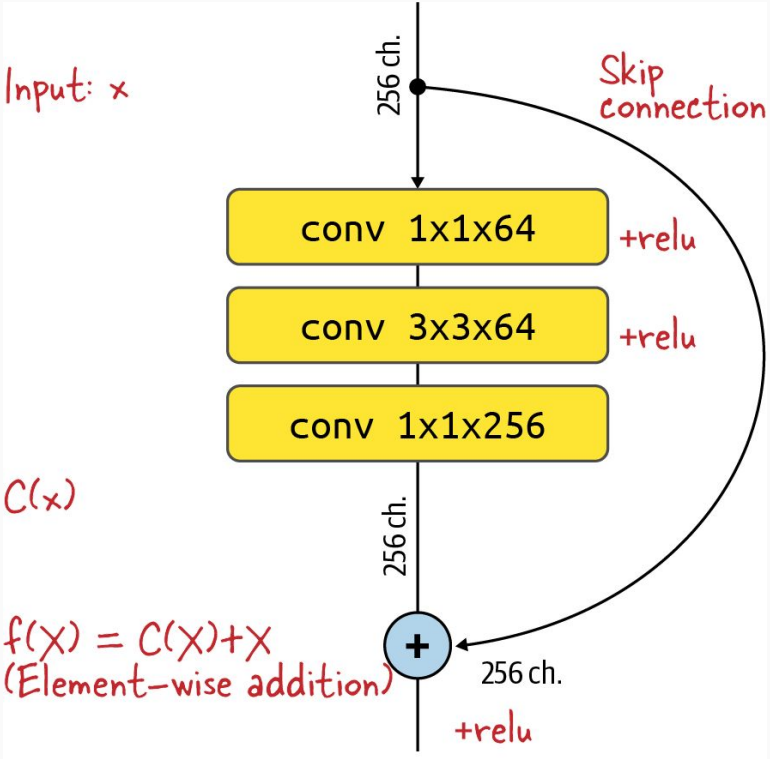
- They tend to converge badly because of vanishing or exploding gradient problems.
- During training, a neural network sees what error (or loss) it is making and tries to minimize this error by adjusting its internal weights.
- It is guided in this by the first derivative (or gradient) of the error.
- Unfortunately, with many layers, the gradients tend to be spread too thin across all layers and the network converges slowly or not at all.



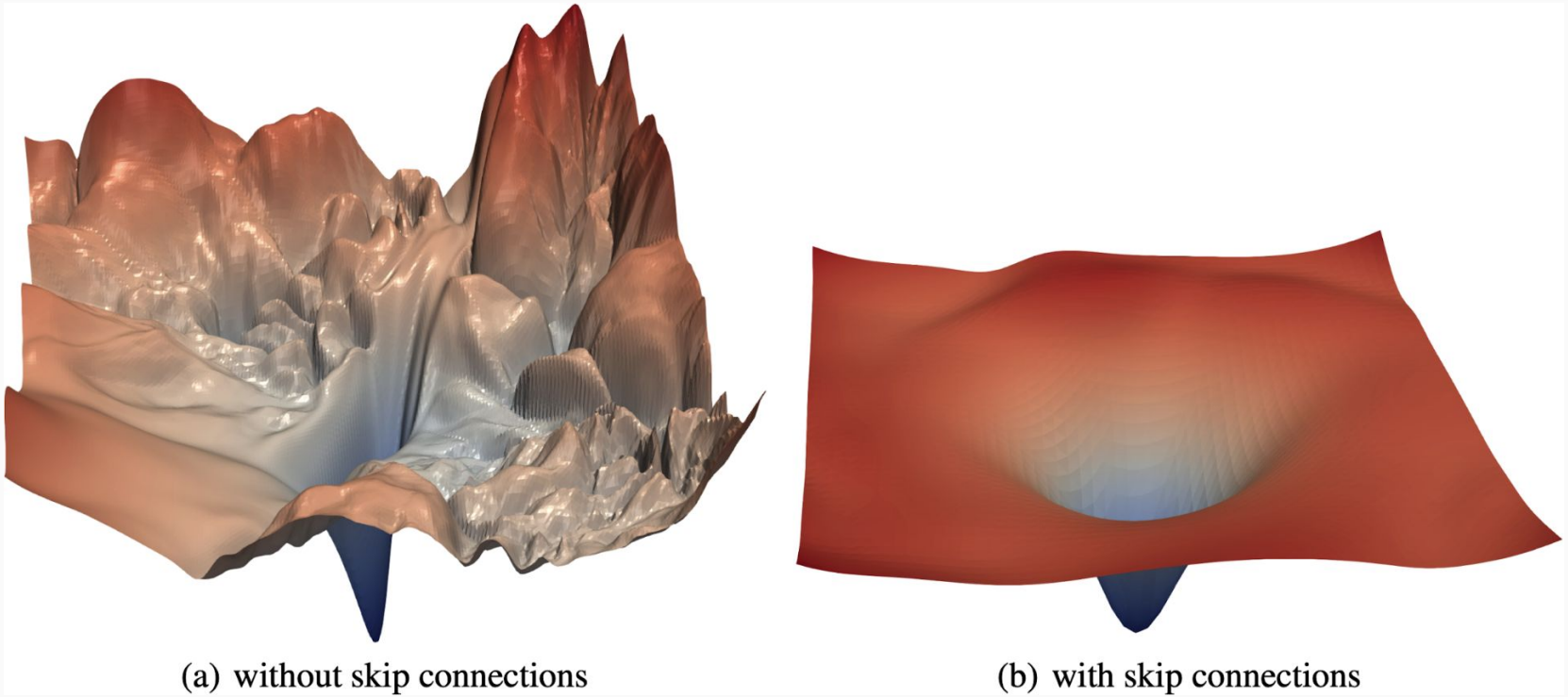
Vanishing gradients - DCGAN. The generator is frozen after 1, 10, and 25 epochs, and the discriminator is trained further. The gradient of the generator decreases rapidly (note log scale); if the discriminator becomes too accurate, the gradients for the generator vanish. Arjovsky & Bottou (2017).

Skip Connections

- Skip connections convey the signal as is, then recombine it with the data that has been transformed by one or more convolutional layers.
- The combining operation is a simple element-by-element addition.
- The output of the block *f(x)* is the sum of the output of the convolutional path *C(x)* and the skip connection *(x)*.
- The convolutional path is trained to compute *C(x) = f(x) − x*, the difference between the desired output and the input.



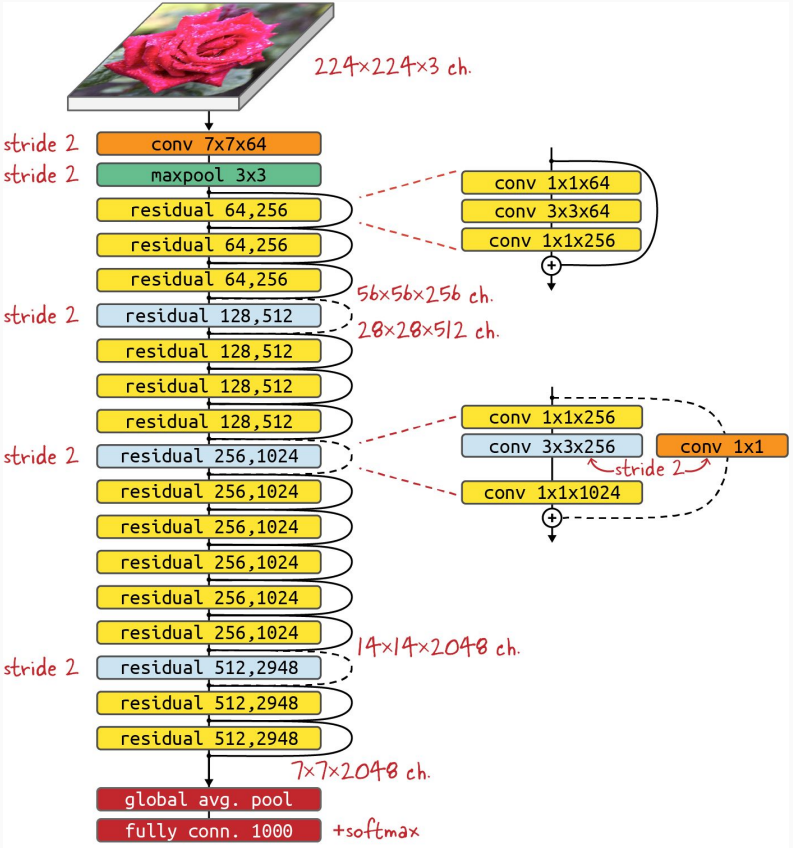
Skip Connections



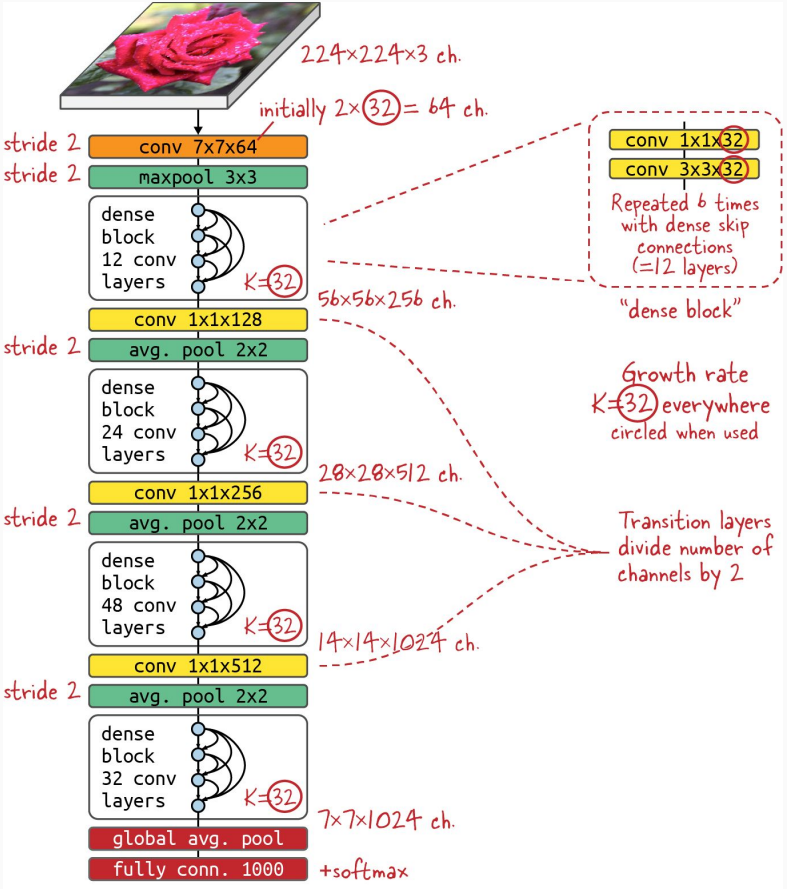
The loss landscape of a 56-layer ResNet as visualized through the “filter normalization scheme” of Li et al. Adding skip connections makes the global minimum much easier to reach. Image from Li et al., 2017.

ResNet

- The ResNet architecture can be instantiated with various depths by stacking more and more residual blocks.
- The convolutional layers are ReLU-activated and use batch normalization.
- A network with this architecture can grow very deep— 50, 100, or more layers.
- Skip connections seem to help gradients flow through the network during the optimization (backpropagation) phase.

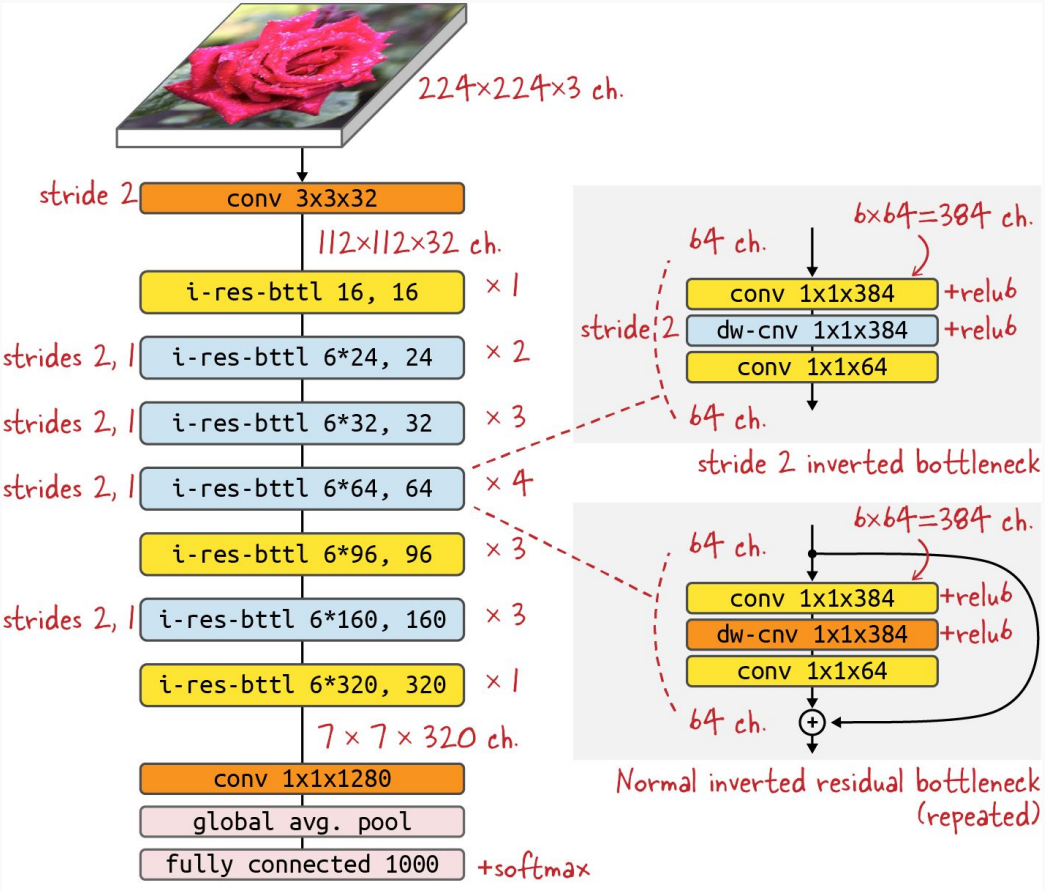


DenseNet



- Convolutional layers produce 32 channels of output, apart from the 1x1 convolutions used as transitions between dense blocks, which are designed to halve the number of channels.
- All convolutions are ReLU-activated and use batch normalization.

MobileNetV2



- The MobileNetV2 architecture, based on repeated inverted residual bottlenecks.
- Repeat counts are in the center column. “conv” indicates regular convolutional layers, while “dw-cnv” denotes depthwise convolutions.

Key Learnings from Vision

Need for Pretrained Models

- Extract and encode visual features from huge image datasets (e.g., ImageNet)
- Models include MobileNet, Xception, ResNet - millions of parameters.
- Learn to represent semantic patterns and concepts in image embedding space
- Not feasible to train large highly-parametric models from scratch on small datasets

Transfer Learning

- Involves taking a pre-trained model designed for one task and reusing it for a different task.
- For example, taking an image classification model trained on ImageNet and using it for a different image classification task with different classes.
- Typically only the fully connected/output layer is changed and retrained.
- The weights of the pretrained layers are frozen as they contain more general features useful for the new task.
- This requires much less data and compute than training a full model from scratch.

How does Transfer Learning work?

- Remove the final 1 to 2 "classification" layers from pretrained model
- These layers compose the "prediction head", designed for specific classes
- Remaining model is a feature extraction network
- Will be "pretrained" with visual concepts from large dataset

Transfer Learning Workflow

1. Load pretrained model
2. Remove prediction head
3. Add small custom classification model on top
4. Freeze pretrained feature extraction weights
5. Only train the newly added classification layers

Benefits and Tradeoffs

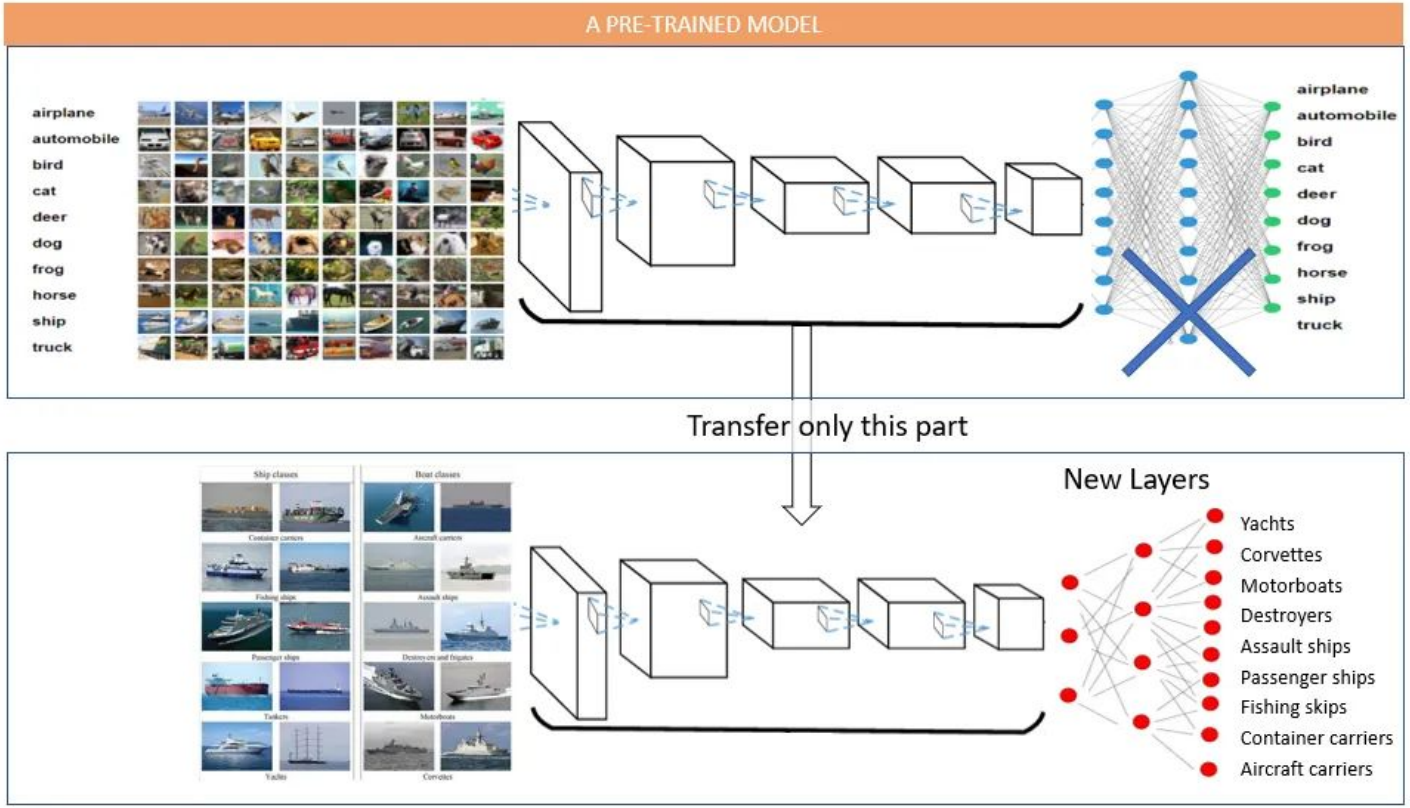
Benefits:

- Leverage rich visual knowledge from large-scale pretraining
- Avoid overfitting from huge model on small target dataset
- Can achieve high performance with little data

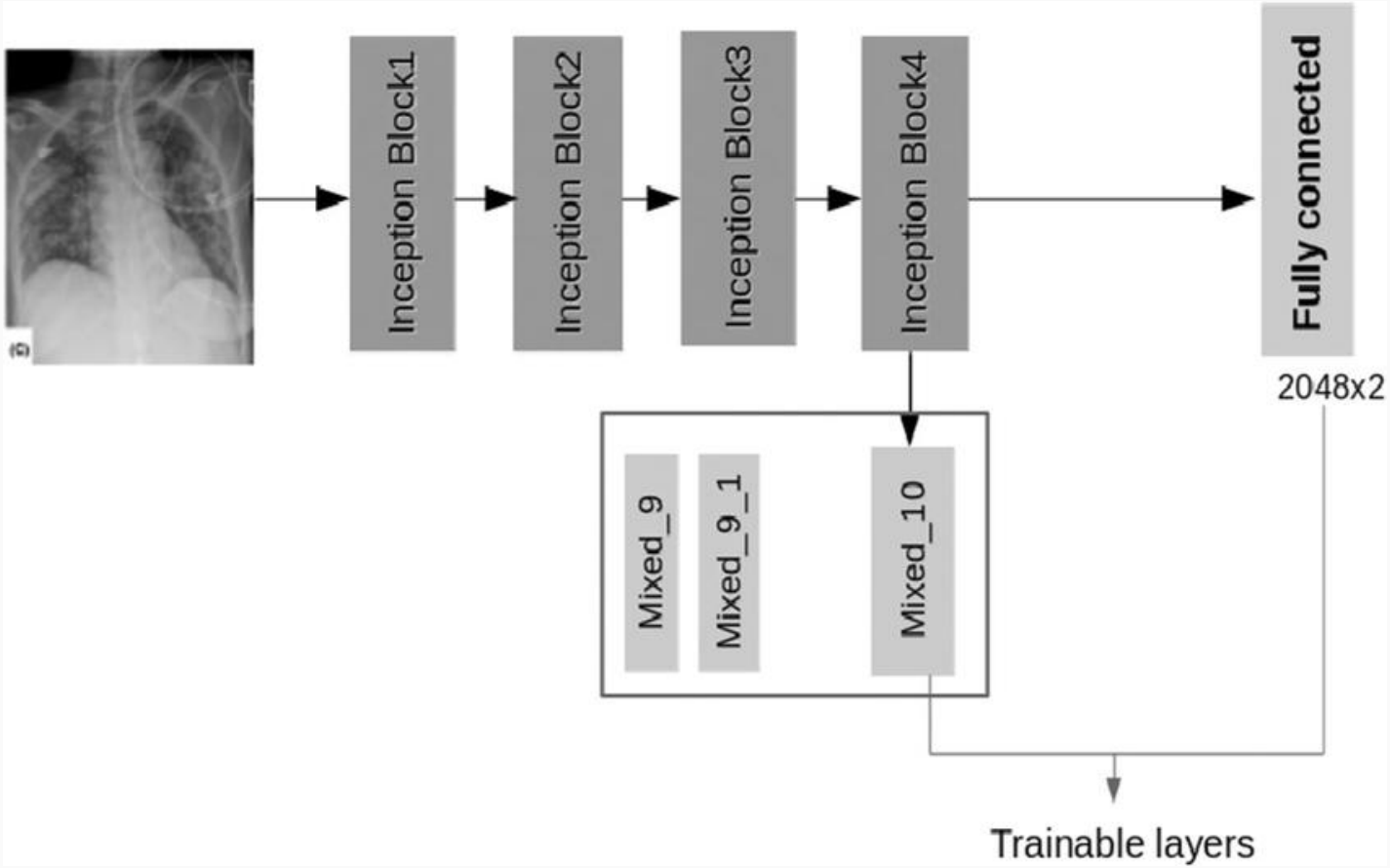
Trade Offs:

- Limited specialization to target domain

Transfer Learning



Transfer Learning - Real World Scenario



Guefrechi, S., Jabra, M. B., Ammar, A., Koubaa, A., & Hamam, H. (2021). Deep learning based detection of COVID-19 from chest X-ray images. *Multimedia Tools and Applications*, *80*, 31803-31820.

Fine-Tuning

- Also starts with a pretrained model, but additionally trains the weights of the entire model on the new dataset and task, not just the output layer.
- Doing full fine-tuning of all layers requires more data and compute than transfer learning, but can adapt the pretrained features better to the new task and achieve higher performance.
- Often, part of the model is still frozen for efficiency and to prevent overfitting.
- For example, just fine-tuning the higher layers related to the task while leaving the early feature layers mostly untouched.

What does Fine-Tuning do?

- Adapt pretrained weights to target dataset by updating weights via backprop
- Specialize the pretrained representations toward target domain
- Achieve even higher performance within target domain

Challenges:

- Learning rates too high: Destroy pretrained knowledge
- Learning rates too low: Very slow training convergence

Techniques for Stable and Effective Fine-Tuning

1. Learning rate ramp-up schedule: Low initial, increasing ramp
2. Differential learning rates for different layers
3. Low LR for early layers, higher for later layers

Transfer Learning Vs. Fine Tuning

- Transfer learning leverages pretrained weights with minimal modification while fine-tuning further optimizes the full model to the new dataset and task at hand.
- Fine-tuning adapts more to the target task but requires more data and resources than simpler transfer learning approaches.
- In practice and literature Fine Tuning is considered as a type of Transfer Learning.

Evaluation Metrics

- Precision: Accuracy of positive predictions per class
- Recall: Fraction of class images correctly classified
- F1 Score: Harmonic mean of precision and recall

Convergence of NLP and Vision

Current

1. In the past NLP borrowed ideas from Vision
2. Today Vision is borrowing ideas from NLP

Quo vadis?

1. Progression in Vision
    a. CNN -> Foundational Vision Models -> Transfer Learning
2. Progression in NLP
    a. RNN -> Transformers -> Foundational NLP Models /LLMs -> Transfer Learning

Attention in Vision

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

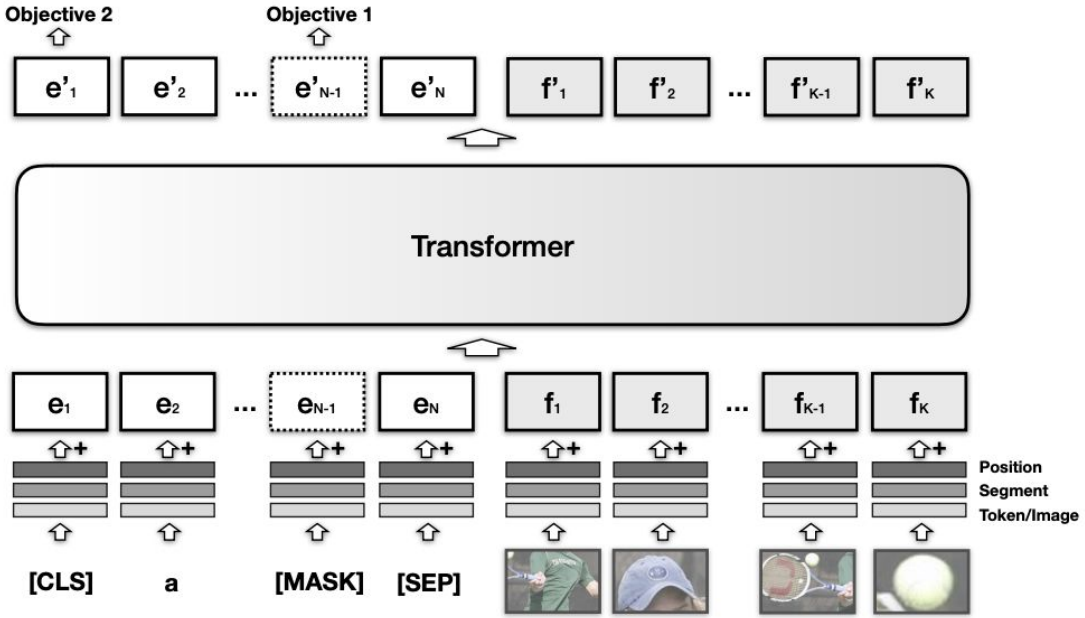
A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

NLP + Vision

A person hits a ball with a tennis racket



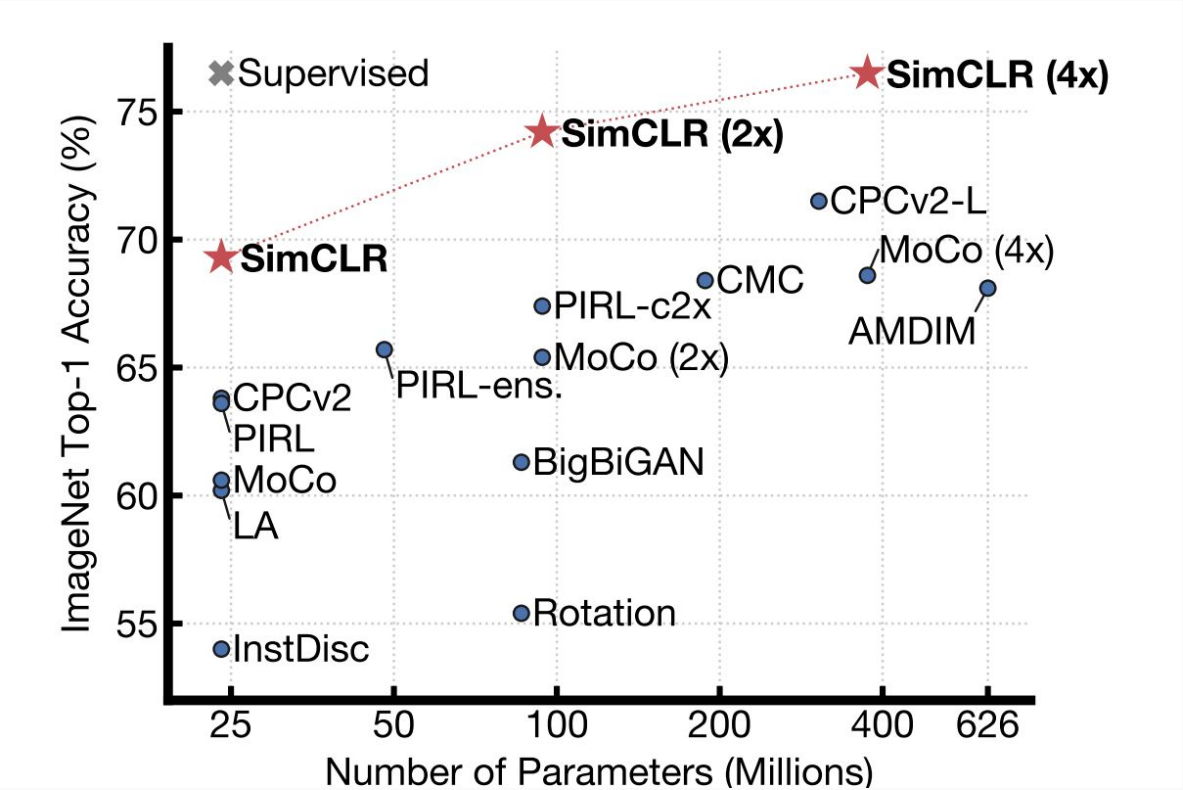
Li, L. H., Yatskar, M., Yin, D., Hsieh, C. J., & Chang, K. W. (2019). Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*.

Contrastive Learning

Contrastive Learning

- Contrastive Learning is a deep learning technique for unsupervised representation learning.
- The goal is to learn a representation of data such that similar instances are close together in the representation space, while dissimilar instances are far apart.
- It has been shown to be effective in various computer vision and natural language processing tasks, including image retrieval, zero-shot learning, and cross-modal retrieval.
- In these tasks, the learned representations can be used as features for downstream tasks such as classification and clustering.

SimCLR



Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020, November). A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (pp. 1597-1607). PMLR.

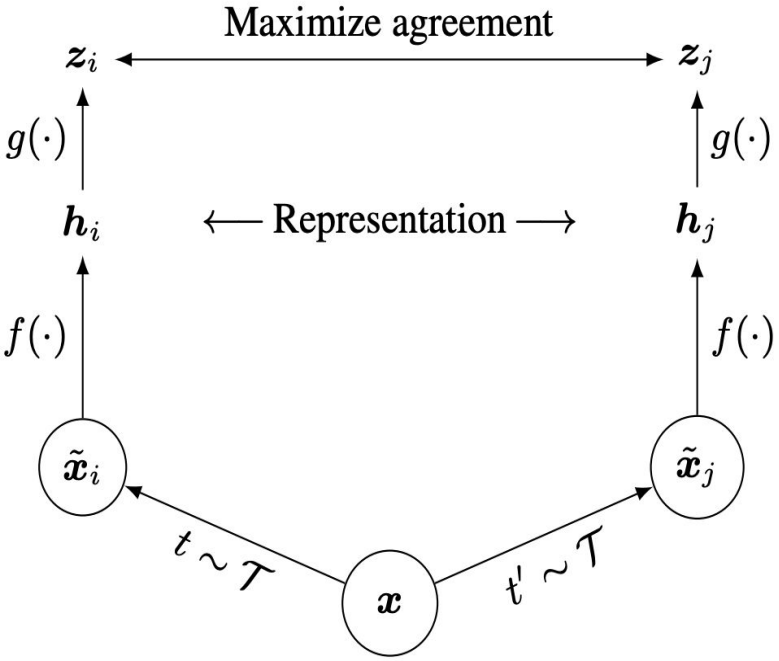
Introduction

- SimCLR, or Simple Framework for Contrastive Learning of Visual Representations, is a groundbreaking approach to self-supervised learning in the domain of computer vision.
- This elegant framework enables the learning of high-quality visual representations without the need for specialized architectures or memory banks, which were common in previous contrastive learning methods.
- SimCLR achieves state-of-the-art performance on the challenging ImageNet dataset, showcasing its effectiveness in learning meaningful and transferable representations.

SimCLR Framework - four main components

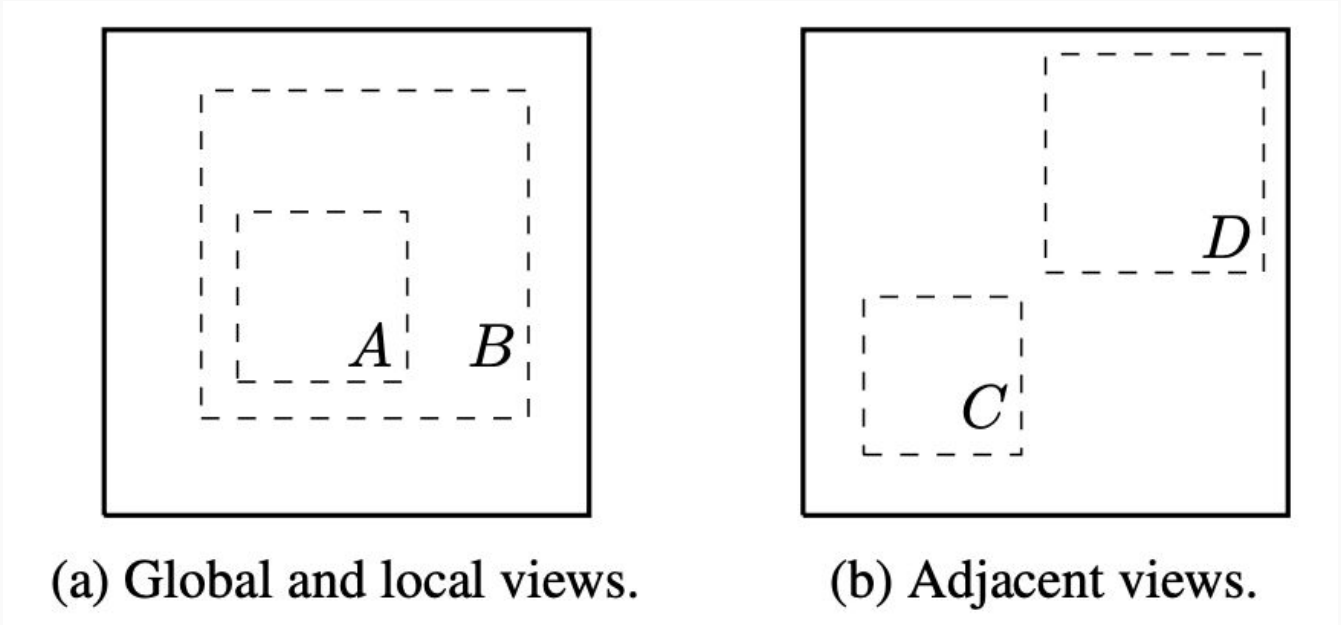
1. First, stochastic data augmentation techniques are applied to generate positive pairs of images, which are different views of the same underlying data sample.
2. These augmented views are then fed into a base encoder network, denoted as f(·), which extracts feature representations from the input images.
3. To map these representations to a space suitable for contrastive learning, a projection head g(·) is introduced.
4. Finally, the contrastive loss function, called NT-Xent (Normalized Temperature-scaled Cross-Entropy), is employed to maximize the agreement between the representations of differently augmented views of the same image, while minimizing the agreement with other images in the batch.

A simple framework for contrastive learning of visual representations.



- Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim T$ and $t' \sim T$) and applied to each data example to obtain two correlated views.
- A base encoder network *f(·)* and a projection head *g(·)* are trained to maximize agreement using a contrastive loss.
- After training is completed, we throw away the projection head *g(·)* and use encoder *f (·)* and representation h for downstream tasks.

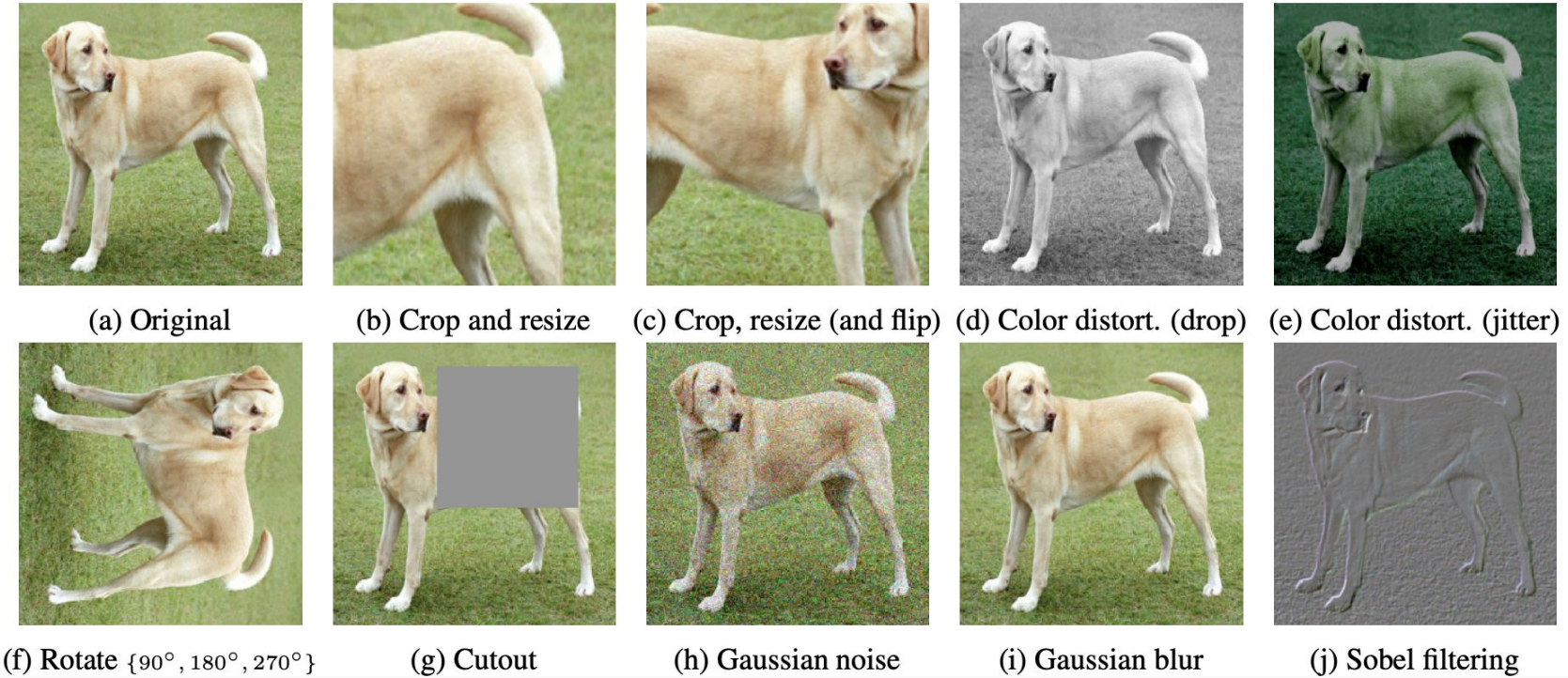
Views



(a) Global and local views. (b) Adjacent views.

- Solid rectangles are images, dashed rectangles are random crops.
- By randomly cropping images, we sample contrastive prediction tasks that include global to local view (B → A) or adjacent view (D → C) prediction.

Illustrations of the studied data augmentation operators.



Each augmentation can transform data stochastically with some internal parameters (e.g. rotation degree, noise level).

Data Augmentation Findings

- One of the key findings of the SimCLR paper is the importance of composing multiple data augmentation techniques.
- The most effective combination found was random cropping followed by random color distortion. Contrastive learning benefits significantly from stronger color augmentations compared to supervised learning methods.
- The combination of random cropping and color distortion helps prevent the network from learning shortcuts and instead focuses on more generalizable features.
- These augmentations play a crucial role in defining the difficulty and quality of the contrastive learning task.

Projection Head

- The incorporation of a learnable nonlinear projection head g(·) between the representation and the contrastive loss is a significant improvement introduced by SimCLR.
- This projection head substantially enhances the quality of the learned representations.
- An interesting finding is that the representation h, which is the output of the base encoder network f(·) before the projection head, is a better representation than the projection z = g(h).
- This suggests that the projection head should be removed after the pre training stage, and the base encoder f(·) should be used to extract the final representations for downstream tasks.

Normalized Cross-Entropy Loss with Adjustable Temperature

- SimCLR introduces the NT-Xent loss function, which is a temperature-scaled cross-entropy loss that operates on the cosine similarity between representations.
- Adjusting the temperature parameter τ in the loss function is crucial for achieving optimal performance.
- The NT-Xent loss encourages the model to learn meaningful representations by pulling together positive pairs while pushing apart negative pairs.
- One advantage of this loss function is that it avoids the need for careful negative mining, which is required by other contrastive loss functions.

Batch Size and Training Steps

- The SimCLR framework benefits significantly from larger batch sizes during training. Increasing the batch size provides a larger number of negative examples, which enhances the contrastive learning process.
- Additionally, SimCLR achieves better performance when trained for longer durations compared to supervised learning methods.
- The paper suggests training for up to 1000 epochs to fully realize the potential of the framework.
- To ensure training stability with large batch sizes and the LARS optimizer, the authors recommend using a square root learning rate scaling instead of the more common linear scaling.

Encoder Architecture

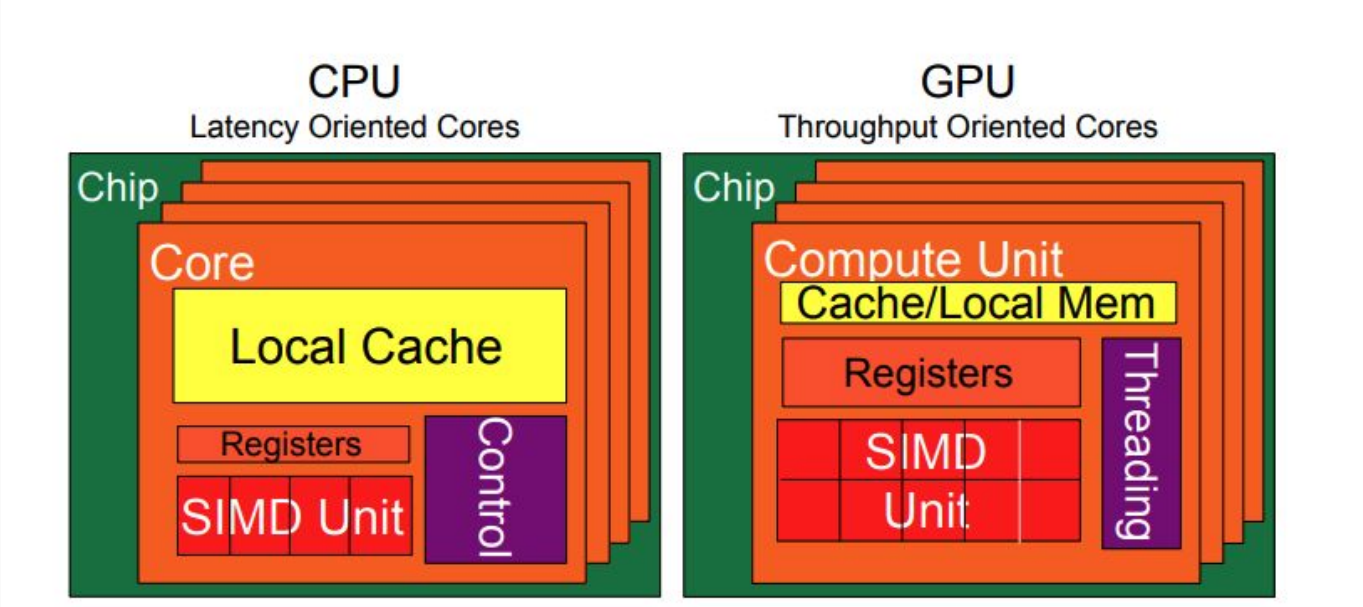
- The choice of encoder architecture plays a vital role in the quality of the learned representations.
- SimCLR demonstrates that wider and deeper encoders lead to improved representation quality.
- Interestingly, contrastive learning benefits even more from increased network width and depth compared to supervised learning methods.
- This finding highlights the potential of contrastive learning to effectively utilize larger and more powerful network architectures.

Results

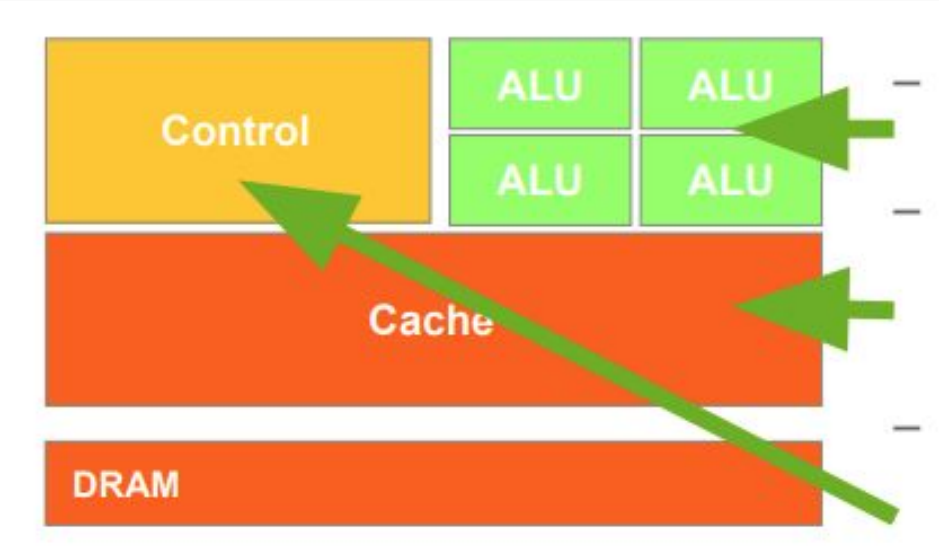
- Semi-Supervised Learning Results
  - SimCLR achieves state-of-the-art results in semi-supervised learning on the ImageNet dataset.
  - By using only 1% of the labeled data and a ResNet-50 encoder, SimCLR attains an impressive top-5 accuracy of 75.5%.
- Transfer Learning Results
  - The representations learned by SimCLR exhibit strong transfer learning performance across a wide range of datasets.
  - When compared to supervised pre-training, SimCLR matches or even exceeds the performance on many datasets.
  - The best results are obtained using a larger ResNet-50 (4x) encoder, indicating that contrastive learning benefits more from increased model capacity compared to supervised learning.
  - These transfer learning results showcase the generalizability and versatility of the representations learned by SimCLR.

Graphical Processing Units

CPU vs GPU


CPU
Latency Oriented Cores
Chip
Core
Local Cache
Registers
SIMD Unit
Control
GPU
Throughput Oriented Cores
Chip
Compute Unit
Cache/Local Mem
Registers
SIMD Unit
Threading

GPU - in detail

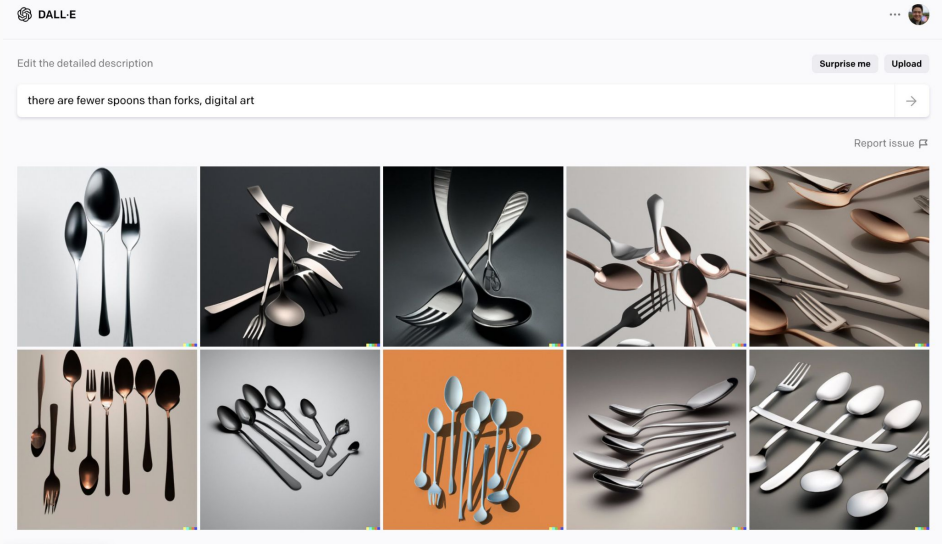


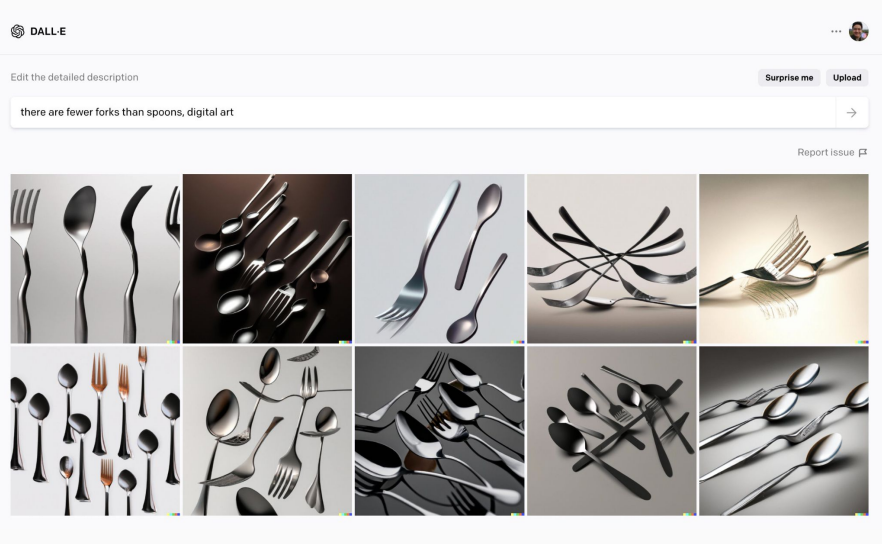
Powerful ALU – Reduced operation latency

Large caches – Convert long latency memory accesses to short latency cache accesses

Future: Is Deep Learning the ultimate solution?

Understanding Intent





Credit - Evan Morikawa

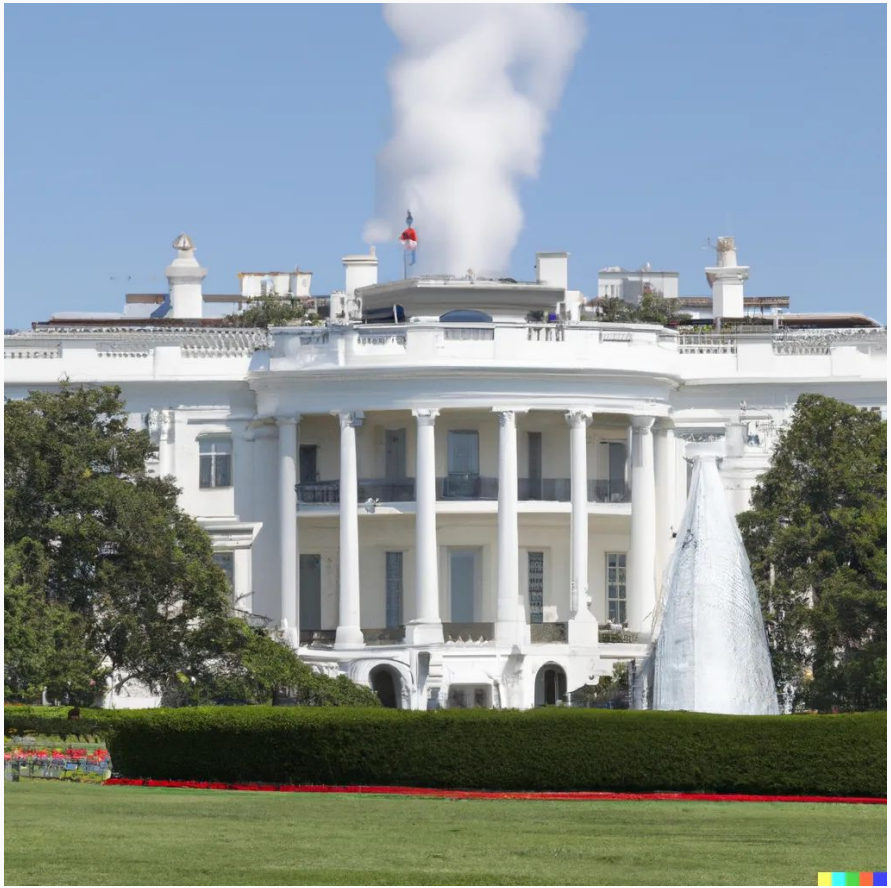
Bias

Flight Attendant

Construction Worker

Credit: OpenAI

Disinformation



Smoke + White House

Credit: OpenAI

Recap

1. Deep Learning in Vision
    a. CNN
    b. Foundational Models in Vision
    c. Key Learnings from Vision
2. Convergence of Deep Learning in NLP and Vision
3. Need Semi-supervised Methods for Data - Contrastive Learning
4. Future

References

1. Lakshmanan, V., Görner, M., & Gillard, R. (2021). *Practical machine learning for computer vision*. " O'Reilly Media, Inc." - Chapter 3
2. Foster, D. (2022). *Generative deep learning*. " O'Reilly Media, Inc.". - Chapter 2
3. Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020, November). A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (pp. 1597-1607). PMLR.

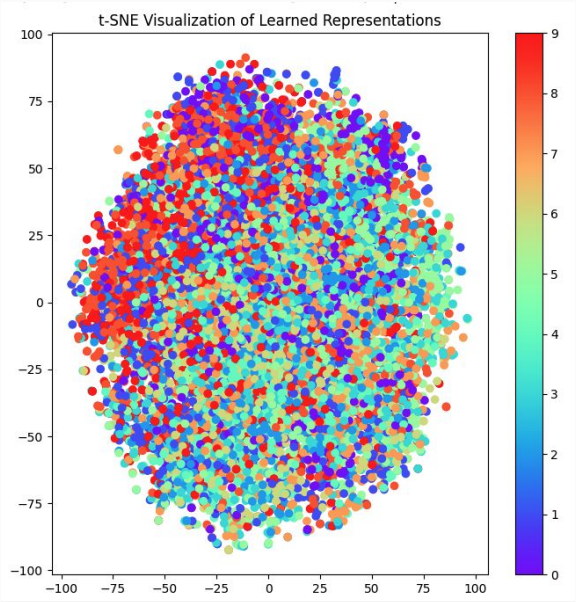
This week

1. Read the SimCLR paper
    a. in less than 200 words discuss at least one-use case where you would need to generate such data to train your model
    b. and comment about the organization of the paper.
2. Pick 2 Class-2-CL-* notebooks, change one of the hyperparameters (e.g. num of epochs).
    a. Either at the beginning of the notebook or at the end make a text cell and in few lines list the hyperparameter change you made and note the changes in the results.

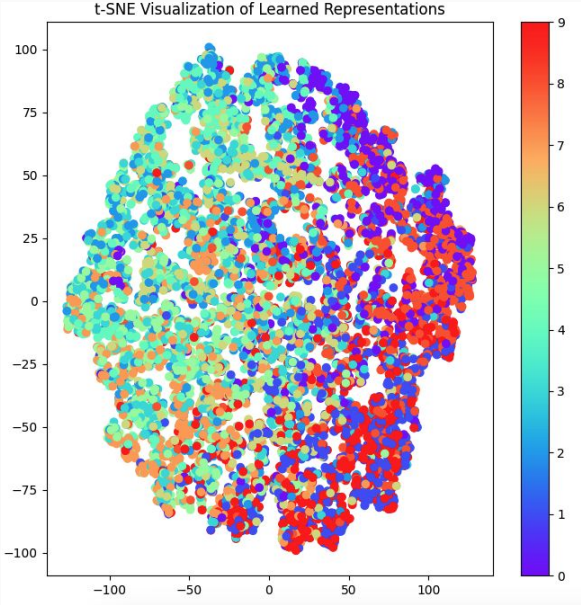
Github Repo for the class: https://github.com/vijaygwu/SEAS8525/

Homework - Contrastive Learning

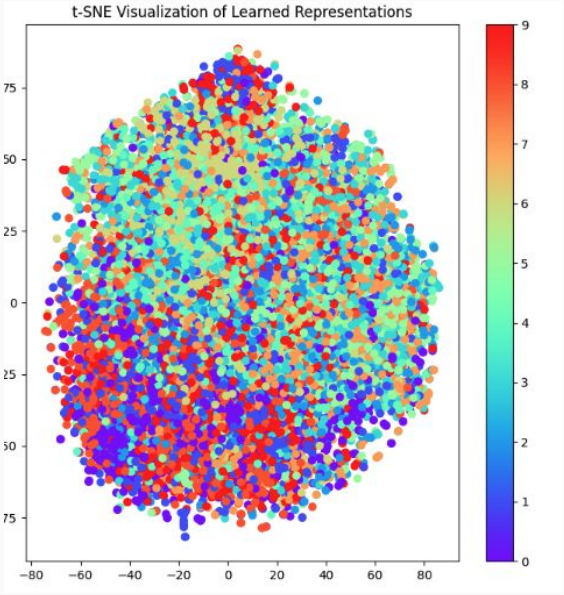
CL_Simple.ipynb



CustomeCLFunction.ipynb



CL_SimpleOptimization.ipynb



Optimizing the Latent Space of Generative Networks

Vijay Raghavan

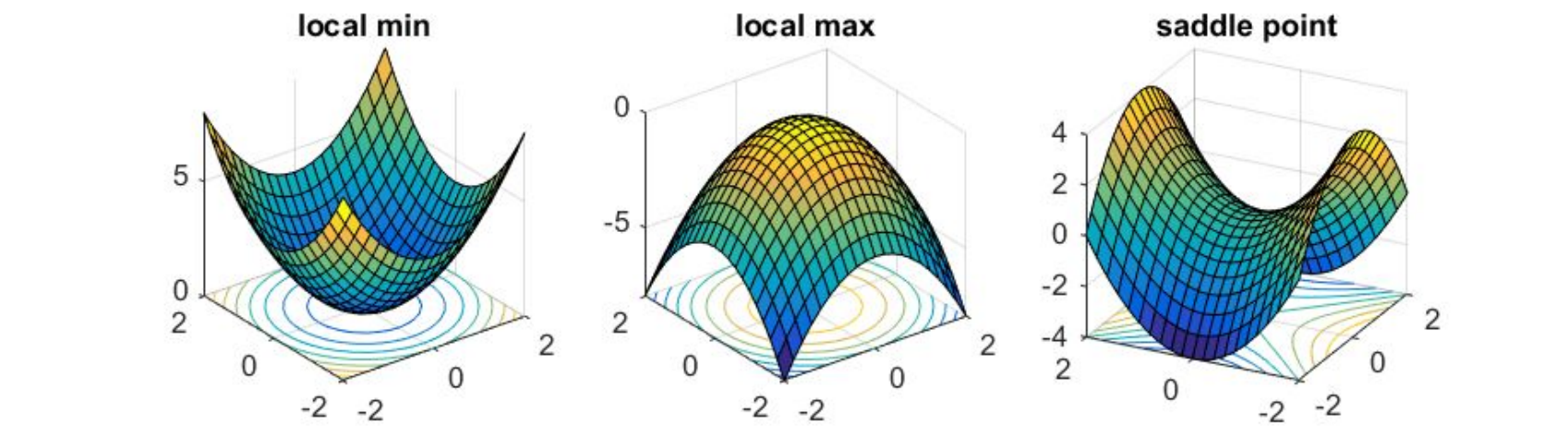
GANs

GANs have been successful in generating realistic natural images

GANs rely on:

1. Saddle point optimization - adversarial game between generator and discriminator
2. Deep convolutional neural networks for generator and discriminator

Saddle Point



Challenges in Training and Evaluating GANs

- Despite their success, GANs are notoriously difficult to train and evaluate
- Adversarial optimization in GANs is sensitive to:
  - Random initialization
  - Architectural choices
  - Hyper-parameter settings
- Advances in stabilizing GAN training have been made, but it remains more art than science
- Discriminator evaluation is difficult due to poor feature transferability
- Application of GANs to non-image data has been relatively limited

Disentangling the Success Factors of GANs in Modeling Natural Images

- Hypothesis: The success of GANs comes from two complementary sources:
  - (A1) Leveraging the powerful inductive bias of deep convnets
  - (A2) The adversarial training protocol
- This work aims to disentangle the factors of success (A1) and (A2) in GAN models
- Proposed algorithm:
  - Relies on (A1): Using deep convnets
  - Avoids (A2): Does not use adversarial training
- The proposed algorithm achieves competitive results compared to GANs

How it Works

1. Start with a large set of images and create a set of random vectors.
2. Pair each image with a corresponding random vector.
3. Train a generator network to produce images from the random vectors, aiming to minimize the difference between generated and real images.
4. Optimize the generator's parameters and the random vectors simultaneously to find the best match.

Results

- Results show that on many image datasets (CelebA, MNIST, SVHN), GLO reproduces the celebrated properties of GAN generations without the GAN training protocol
- GLO performs worse than GANs on LSUN.