DIRECTORATE OF ONLINE EDUCATION
(A unit of Manipal Academy of Higher Education, Manipal)

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

WELCOME TO THE
UNIVERSITY OF INSPIRED LEARNING
www.manipal.edu

Course	**Computer Vision**
Segment 6	**Deep Learning for Computer Vision**
Faculty	**Prof. Mahesh Inamdar**

Recap

1. Vision-based Navigation Systems
2. Object Detection and Tracking for Robotics
3. SLAM (Simultaneous Localization and Mapping)

Overview of Topics

1. Contrastive Learning
2. Vision Transformer
3. Image Synthesis Techniques

NEURAL NETWORKS AND DEEP LEARNING

Background on Neural Networks

- Networks of nonlinear computing elements (artificial neurons) similar to visual cortex connections in mammals.
- Names: neural networks, neurocomputers, parallel distributed processing models, neuromorphic systems, layered self-adaptive networks, connectionist models.
- Objective: Adaptively learn parameters of decision functions via training patterns.

Pattern Recognition Techniques

- Traditional techniques require human-engineered feature extraction.
- Neural networks learn representations automatically through backpropagation.
- Deep learning refines representations into more abstract levels.

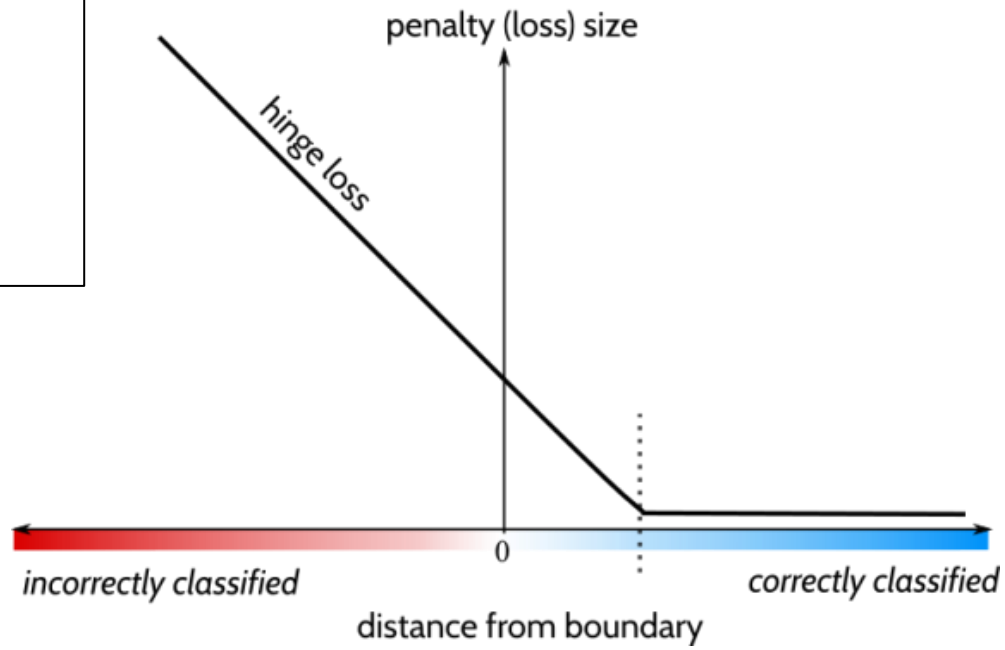
HINGE LOSS FUNCTION

MULTI-CLASS SVM LOSS (Soft Margin)

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Hinge Loss not only penalizes the wrong predictions but also the right predictions that are not confident.



LOSS FUNCTION

QUANTIFES UNHAPPINESS WITH PREDICTIONS (TRAINING SETS)

			
-3.45	-0.51	3.42	PLANE
2.9	3.58	1.5	DOG
3.2	-2.09	2.13	CAT
-0.23	3.33	1.21	CAR

OPTIMIZATION TO MINIMIZE LOSS FUNCTION

2.9	DOG
3.2	CAT
-0.23	CAR

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

CLASSIFIER SCREWED UP BY 0.7

L = max (0,2.9-3.2+1)

+max (0,-0.23-3.2+1)

max (0,0.7) + max (0,-2.43)

L = 0.7+0=0.7

		
3.58	1.5	DOG
-2.09	2.13	CAT
3.33	1.21	CAR

$$L = \max(0, 3.58 - 3.33 + 1) + \max(0, -2.09 - 3.3 + 1)$$

$$\max(0, 1.25) + \max(0, -5.39)$$

$$L = 1.25 + 0 = 1.25$$

$$L = \max(0, 2.13 - 1.5 + 1) + \max(0, 1.21 - 1.5 + 1)$$

$$\max(0, 1.63) + \max(0, 0.71)$$

$$L = 0.163 + 0.71 = 0.873$$

$$L_{avg} = (0.7 + 1.25 + 0.873)/3 = 0.941$$

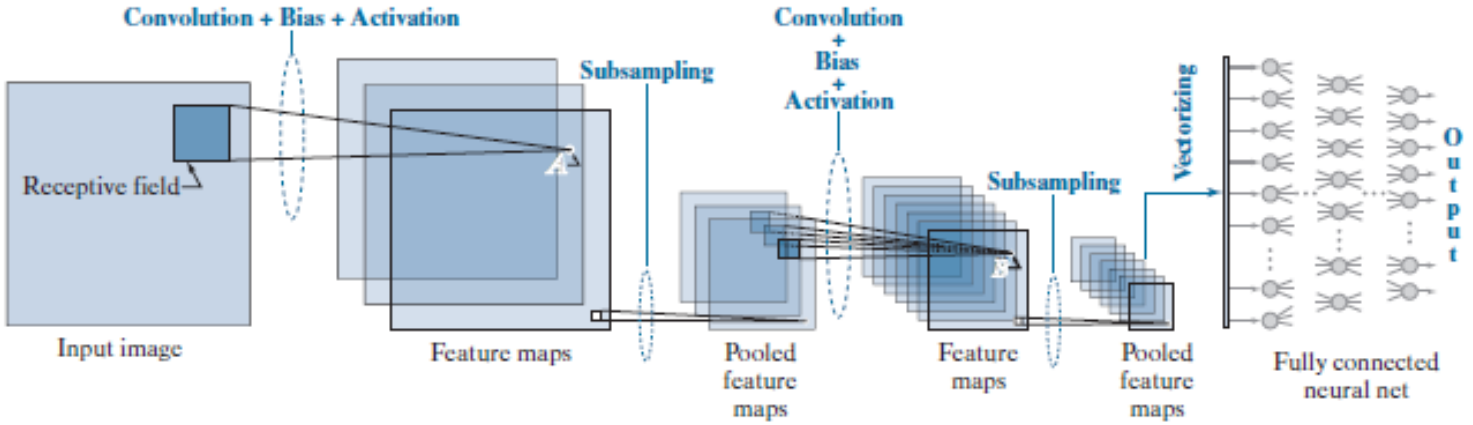
Kullback-Leibler Divergence LOSS FUNCTION

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) \cdot (\log p(x_i) - \log q(x_i))$$

DEEP CONVOLUTIONAL NEURAL NETWORKS

Basics of How a CNN Operates

- Spatial convolution: sum of products between pixels and kernel weights at each spatial location.
- Convolution result at each location is akin to a neuron output in fully connected neural networks.
- Add bias and pass through activation function for complete analogy.



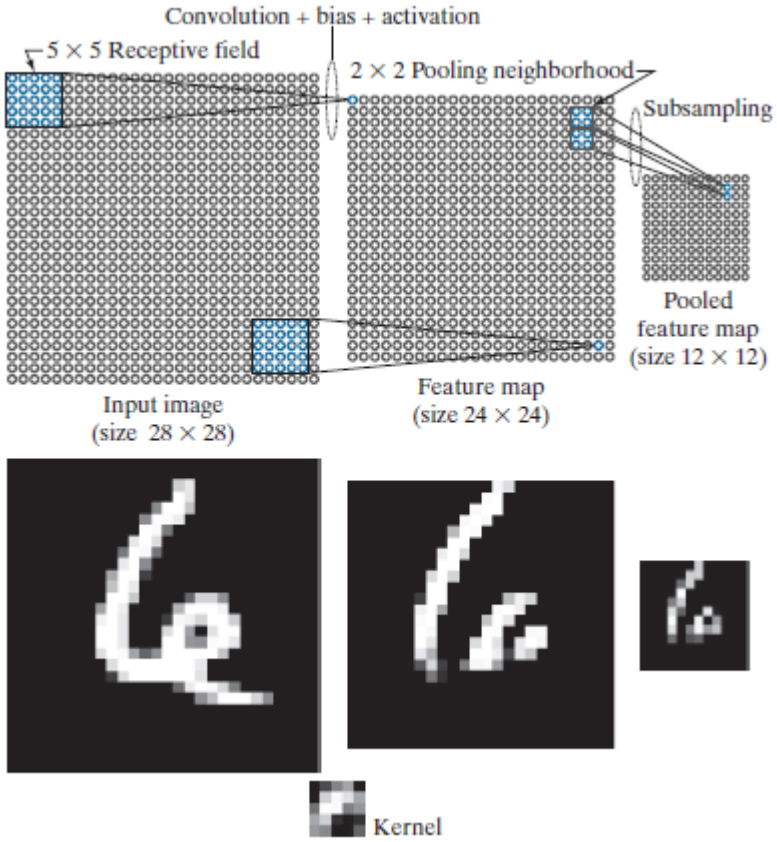
Receptive Field

- Receptive fields: neighborhoods in input image for convolution.
- Convolution values generated by moving receptive field over image with a sum of products operation.
- Stride: number of spatial increments by which receptive field is moved.
- Larger strides reduce data, e.g., stride of two reduces image resolution by half in each spatial dimension.

DEEP CONVOLUTIONAL NEURAL NETWORKS

Feature Maps

- Convolution followed by adding bias and activation function generates single value for each (x, y) location.
- Resulting 2-D array stored as feature map.
- Feature maps extract image features like edges and blobs.
- Weight (parameter) sharing: same weights and bias used across all receptive field locations for feature detection consistency.



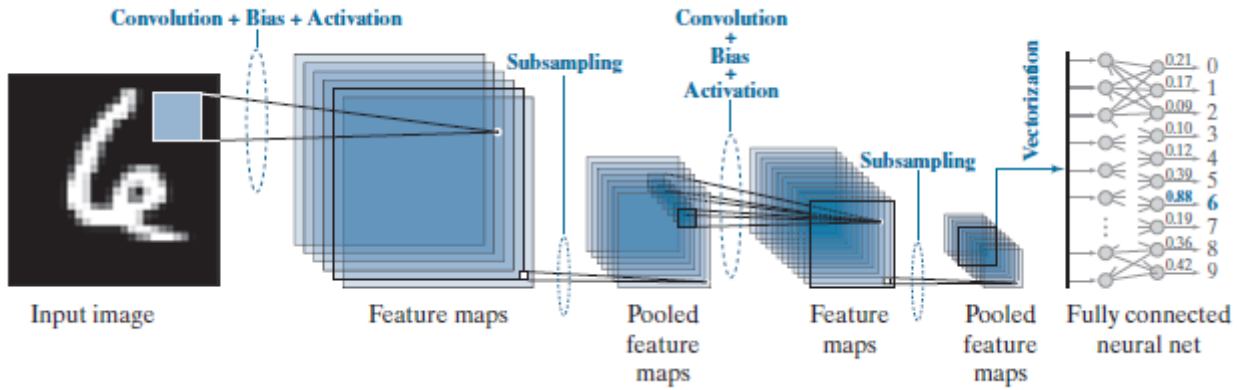
Pooling (Subsampling)

- Pooling reduces spatial resolution for translational invariance and data volume reduction.
- Pooling methods:
  1. Average pooling: replace values with neighborhood average.
  2. Max-pooling: replace values with neighborhood maximum.
  3. L2 pooling: replace values with square root of sum of squares.
- Pooled feature maps are of reduced resolution.

DEEP CONVOLUTIONAL NEURAL NETWORKS

Multi-layer CNNs

- Multiple pooled feature maps from first layer serve as inputs to next layer.
- Convolution, bias addition, and activation repeated for each input with different kernels and biases.
- Outputs combined by superposition.



Classification in CNNs

- Last pooled layer outputs are 2-D arrays, vectorized for fully connected neural network input.
- Vectorized 2-D pooled feature maps concatenated to form a single vector.
- Fully connected neural net processes vector to determine class of input based on highest output value.

DEEP CONVOLUTIONAL NEURAL NETWORKS

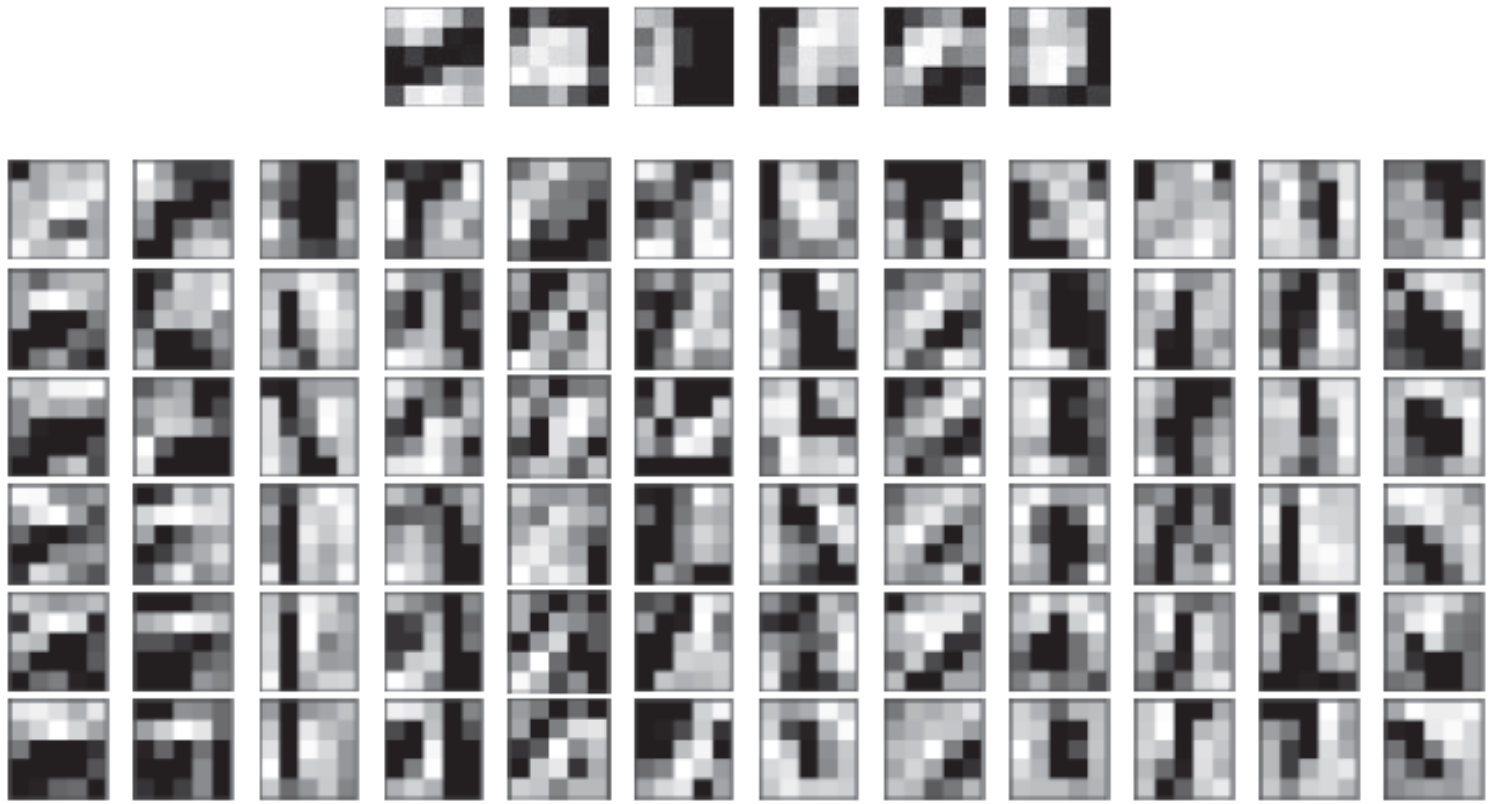
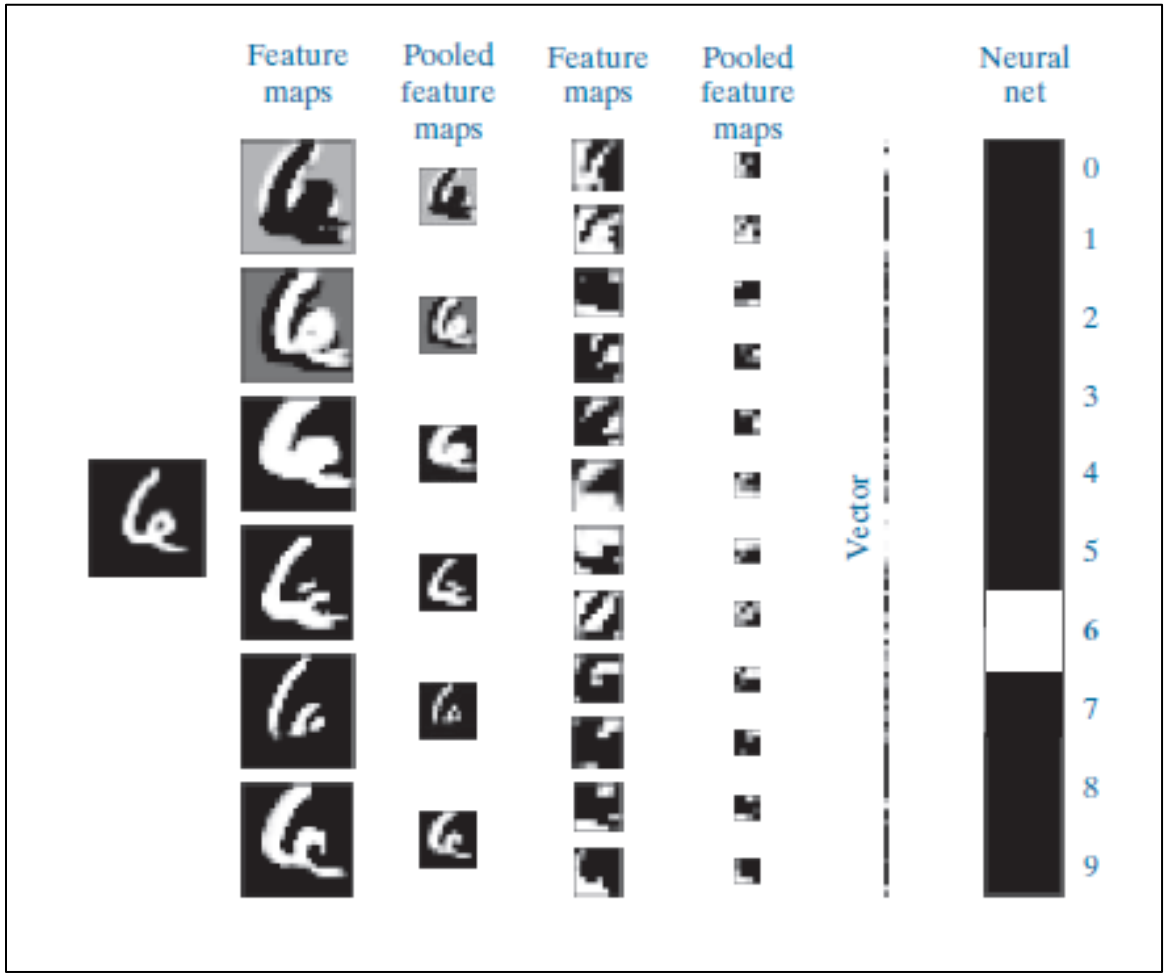
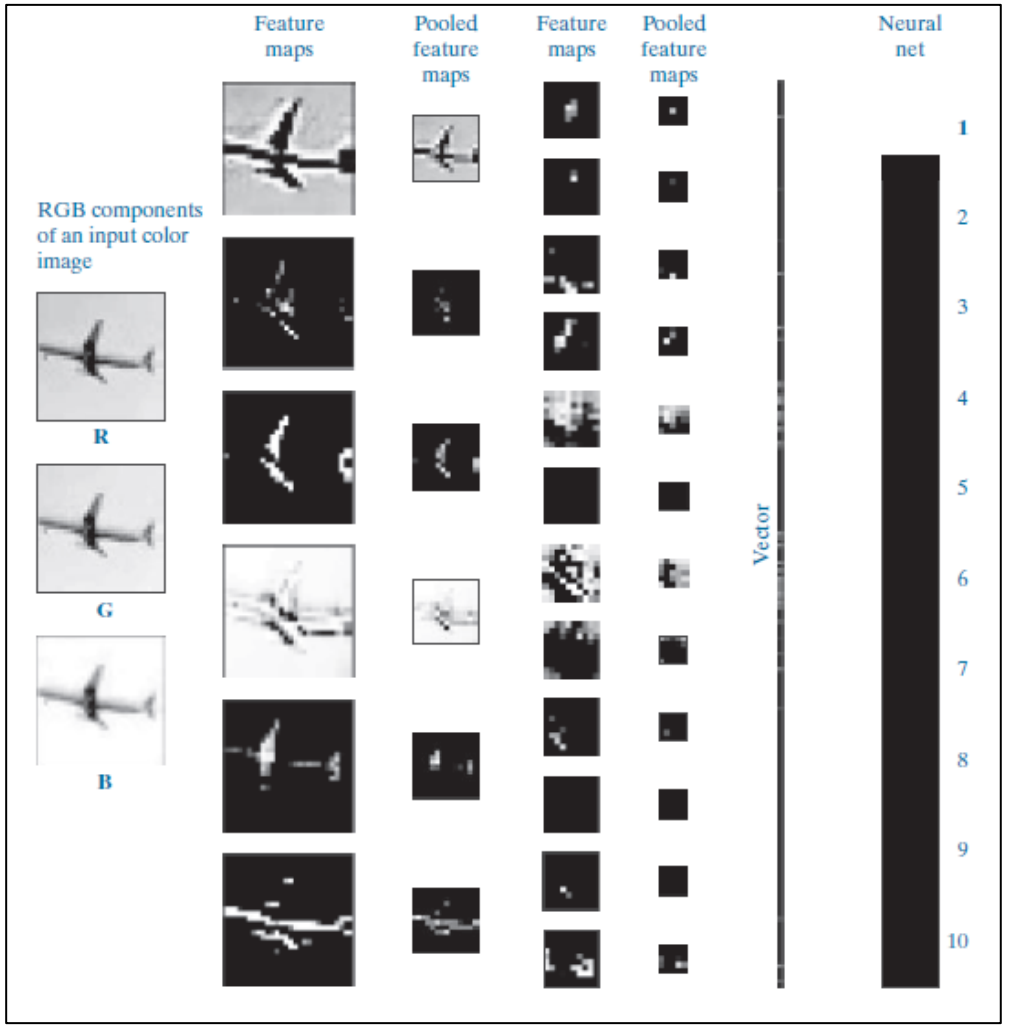


FIGURE 12.43 Top: The weights (shown as images of size 5×5) corresponding to the six feature maps in the first layer of the CNN in Fig. 12.42. Bottom: The weights corresponding to the twelve feature maps in the second layer.

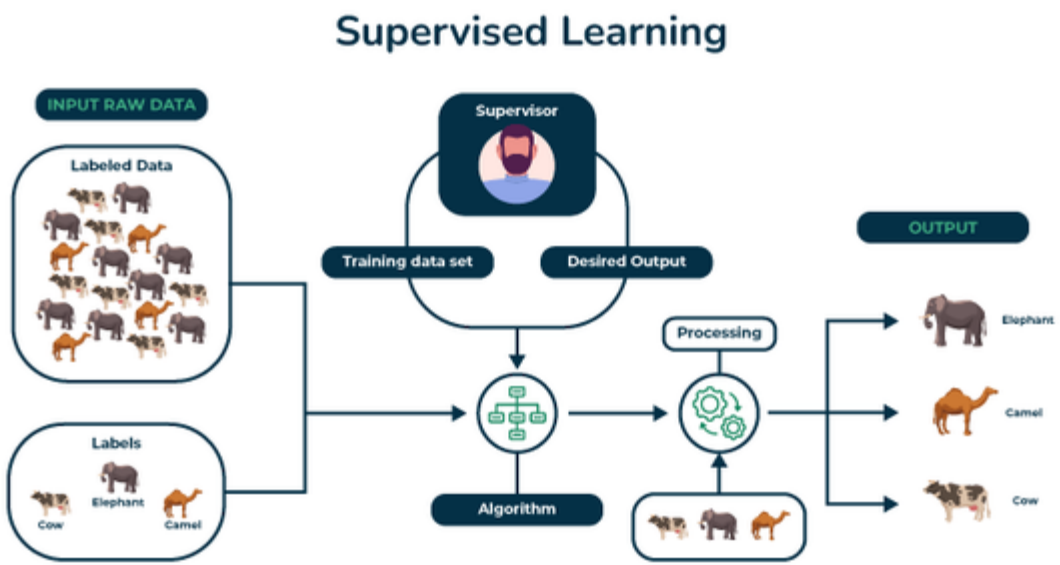
DEEP CONVOLUTIONAL NEURAL NETWORKS





Supervised Contrastive Learning (SCL)

- **Supervised** contrastive learning (SCL) is a branch that uses labeled data to train models explicitly to differentiate between similar and dissimilar instances.
- In SCL, the model is trained on pairs of data points and their labels, indicating whether the data points are similar or dissimilar.
- The objective is to learn a representation space where similar instances are clustered closer together, while dissimilar instances are pushed apart.

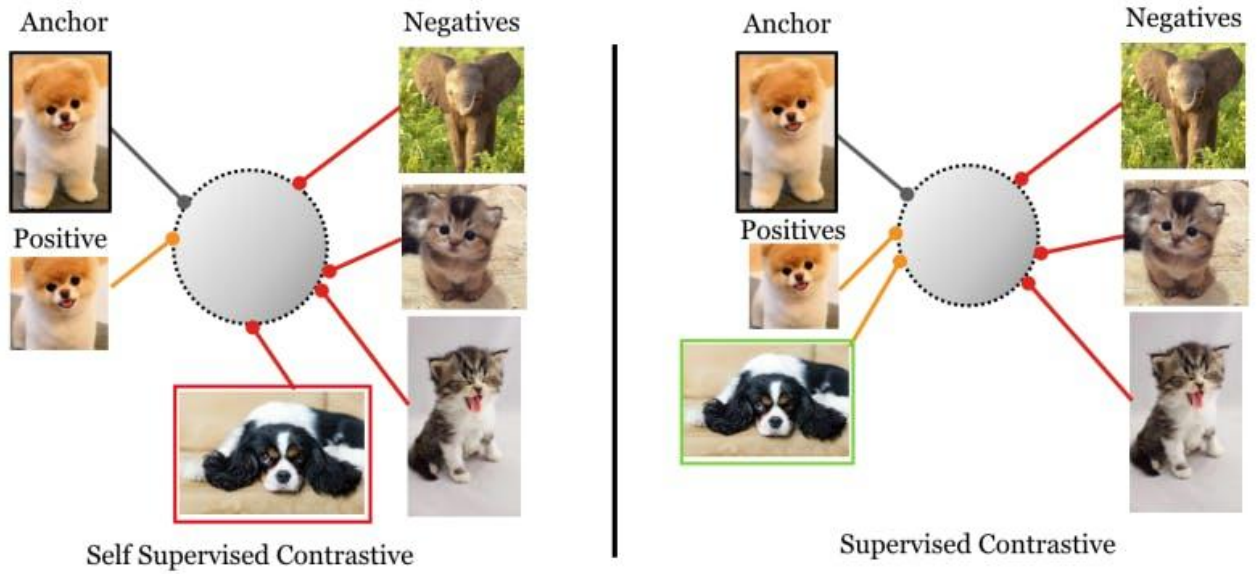


https://lilianweng.github.io/posts/2021-05-31-contrastive/

Supervised Contrastive Learning (SCL)

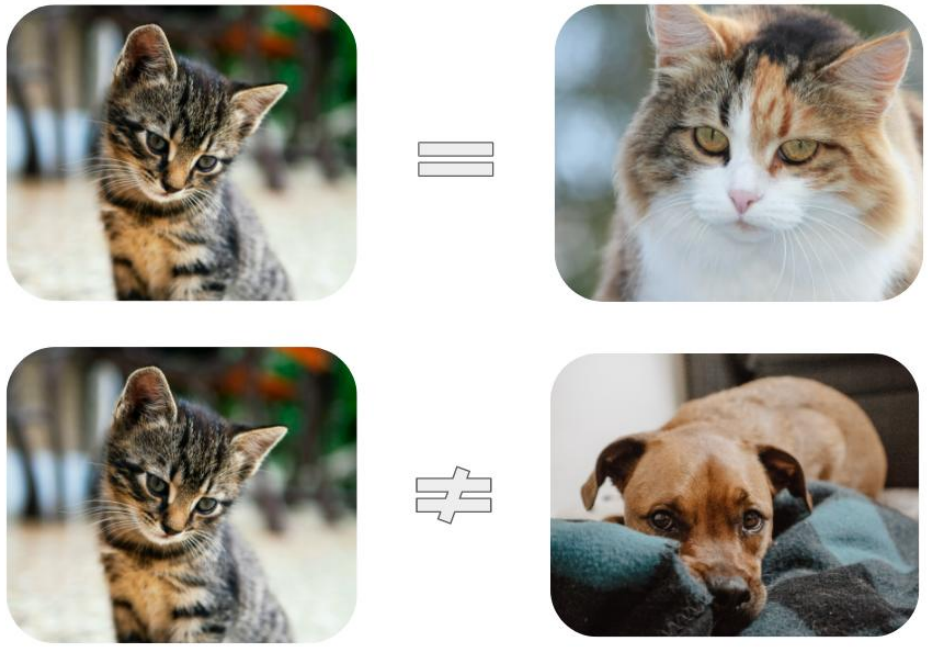
- Self-supervised contrastive learning (SSCL) takes a different approach by learning representations from unlabeled data without relying on explicit labels. SSCL leverages the design of pretext tasks, which create positive and negative pairs from the unlabeled data.

- These pretext tasks are carefully designed to encourage the model to capture meaningful features and similarities in the data.



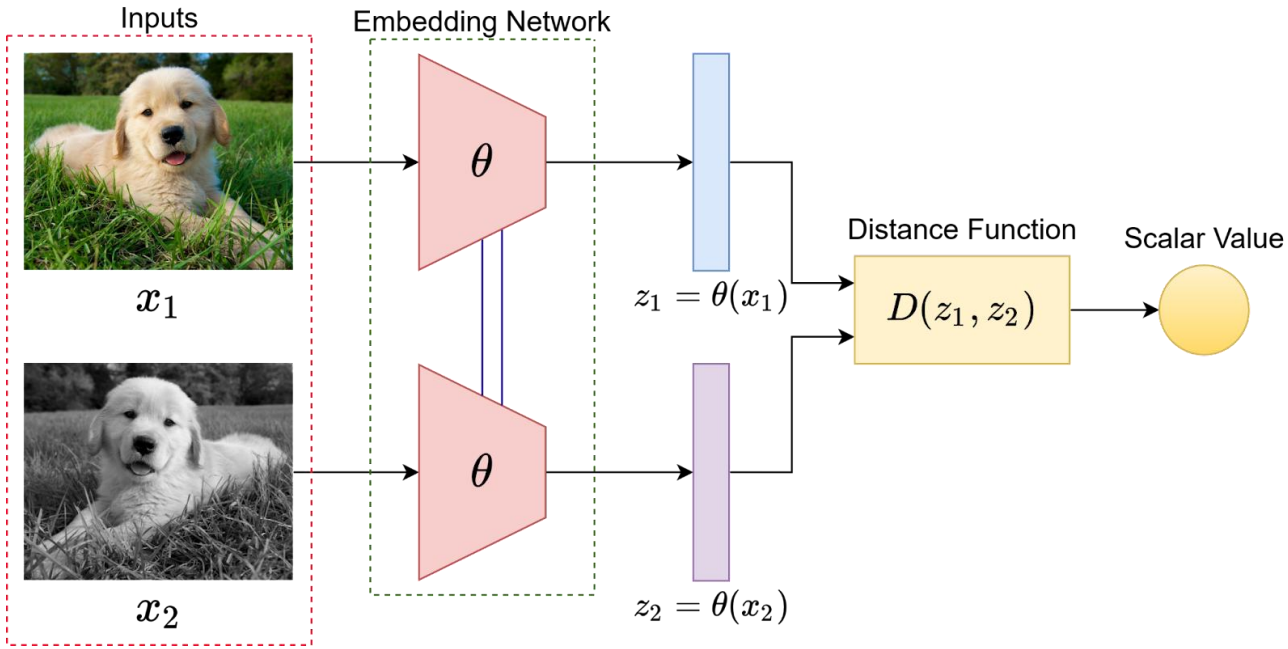
Contrastive Representation Learning

- Contrastive learning is an approach that focuses on extracting meaningful representations by contrasting positive and negative pairs of instances.
- It leverages the assumption that similar instances should be closer in a learned embedding space while dissimilar instances should be farther apart.
- The goal of contrastive representation learning is to learn such an embedding space in which similar sample pairs stay close to each other while dissimilar ones are far apart.
- Contrastive learning can be applied to both supervised and unsupervised settings.
- When working with unsupervised data, contrastive learning is one of the most powerful approaches in self-supervised learning.



How does Contrastive Learning Work?

- Contrastive learning has proven a powerful technique, allowing models to leverage large amounts of unlabeled data and improve performance even with limited labeled data.
- The fundamental idea behind contrastive learning is to encourage similar instances to be mapped closer together in a learned embedding space while pushing dissimilar instances farther apart.
- By framing learning as a discrimination task, contrastive learning allows models to capture relevant features and similarities in the data.



Triplet Loss

Triplet loss was originally proposed in the FaceNet (Schroff et al. 2015) paper and was used to learn face recognition of the same person at different poses and angles.

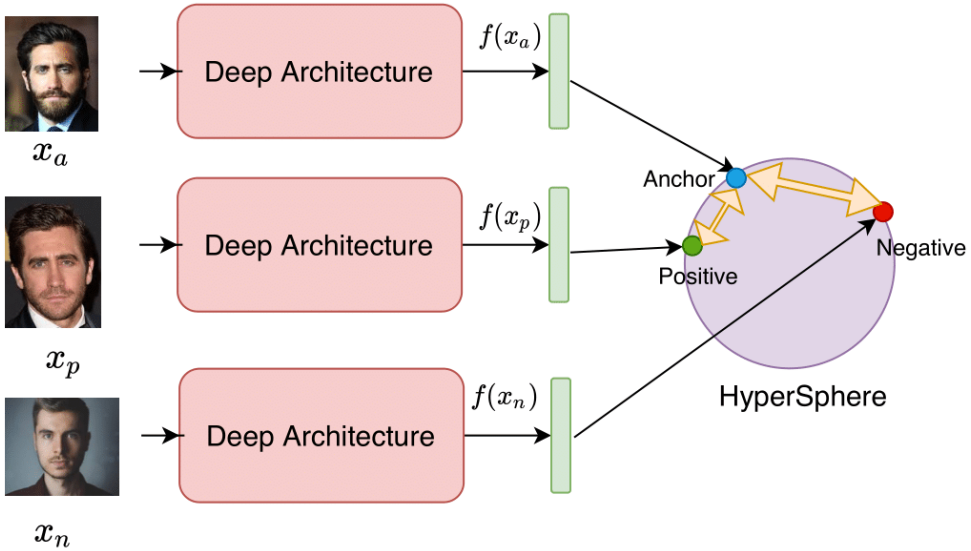
Fig. 1. Illustration of triplet loss given one positive and one negative per anchor.
(Image source: Schroff et al. 2015)

Given one anchor input $\mathbf{x}$, we select one positive sample $\mathbf{x}^+$ and one negative $\mathbf{x}^-$, meaning that $\mathbf{x}^+$ and $\mathbf{x}$ belong to the same class and $\mathbf{x}^-$ is sampled from another different class. Triplet loss learns to minimize the distance between the anchor $\mathbf{x}$ and positive $\mathbf{x}^+$ and maximize the distance between the anchor $\mathbf{x}$ and negative $\mathbf{x}^-$ at the same time with the following equation:

$$\mathcal{L}_{\text{triplet}}(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) = \sum_{\mathbf{x} \in \mathcal{X}} \max\left(0, \|f(\mathbf{x}) - f(\mathbf{x}^+)\|_2^2 - \|f(\mathbf{x}) - f(\mathbf{x}^-)\|_2^2 + \epsilon\right)$$

where the margin parameter ϵ is configured as the minimum offset between distances of similar vs dissimilar pairs.

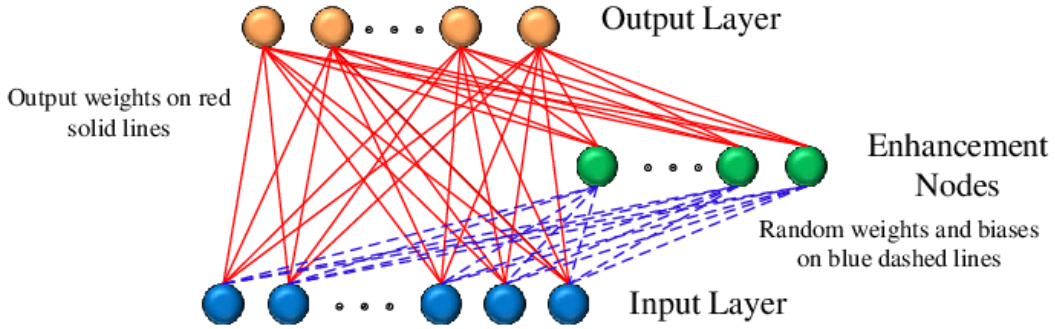
It is crucial to select challenging $\mathbf{x}^-$ to truly improve the model.



Random Vector Functional Unit

RVFL

- RVFL networks are essentially a modified version of the multilayer perceptron (MLP).
- The key difference is that in RVFL, the hidden layer parameters (weights and biases) are randomly assigned and remain fixed, while only the output layer weights are trained.
- This makes the error function quadratic in the output weights, allowing for faster optimization.



RVFL

- **Faster Training**: Since the weights and biases of the hidden layer are fixed and randomly assigned, the training process is significantly faster. Only the output weights need to be trained, which simplifies the optimization problem1.
- **Avoidance of Local Minima**: Traditional dense layers can get stuck in local minima during training. RVFL networks, by fixing the hidden layer parameters, avoid this issue and ensure a more stable training process.
- **Simplified Hyperparameter Tuning**: With fewer parameters to tune, RVFL networks simplify the hyperparameter tuning process, making them easier to implement and optimize.

SimCLR

SimCLR (Chen et al, 2020) proposed a simple framework for contrastive learning of visual representations. It learns representations for visual inputs by maximizing agreement between differently augmented views of the same sample via a contrastive loss in the latent space.

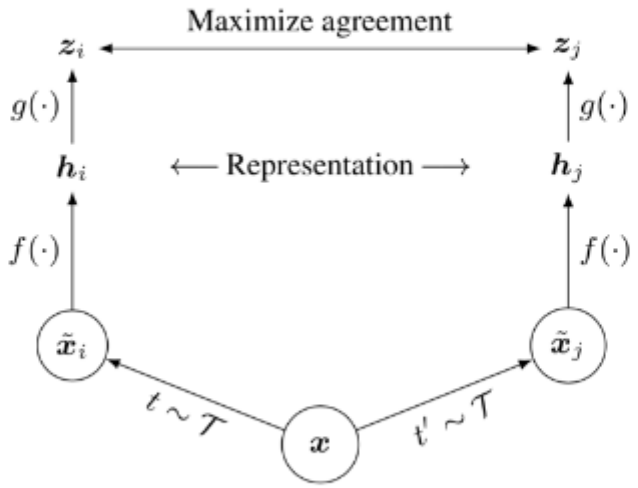
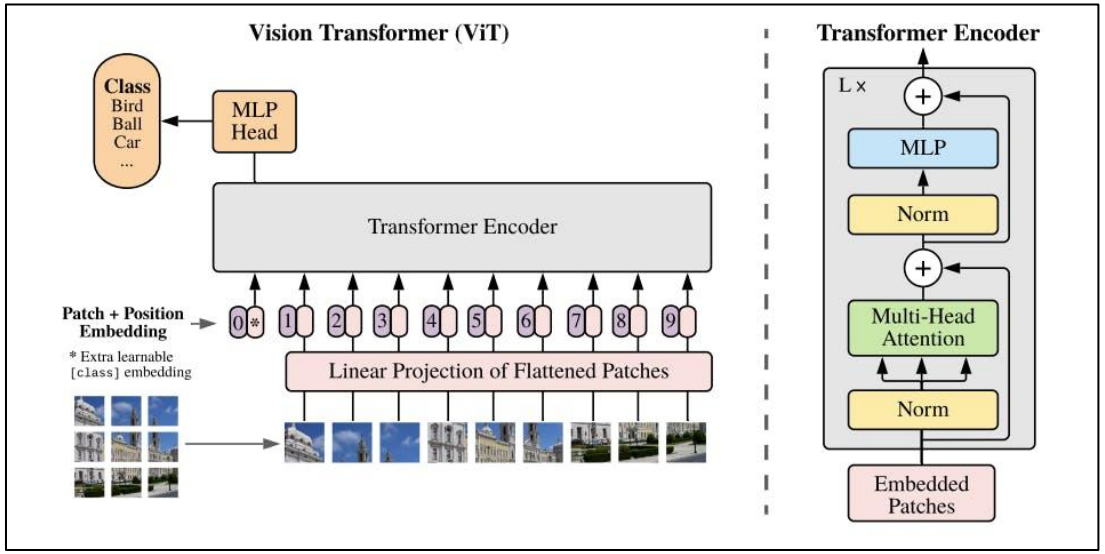


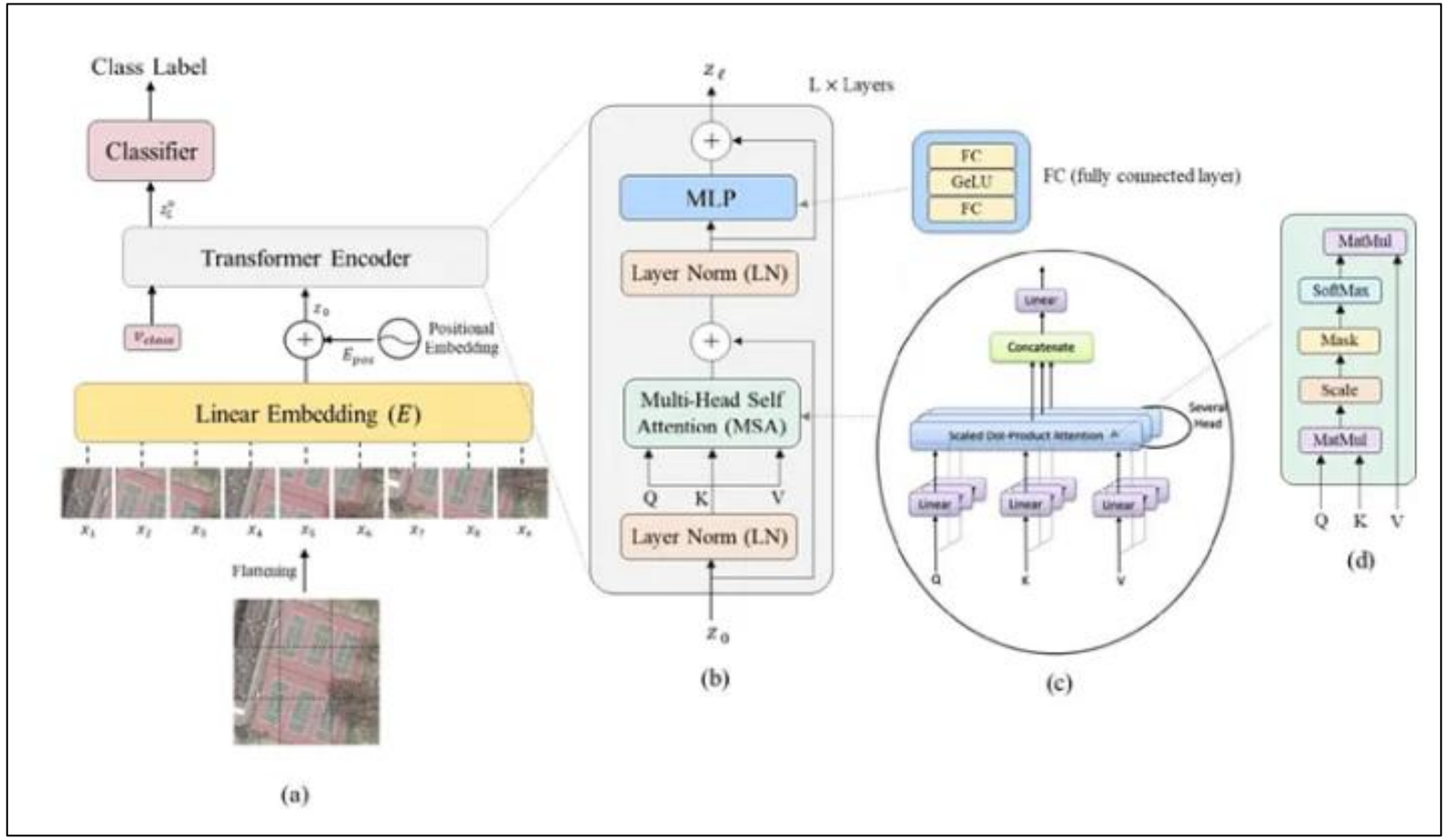
Fig. 6. A simple framework for contrastive learning of visual representations.
(Image source: Chen et al, 2020)

Vision Transformers

- While CNNs have been instrumental in computer vision, a paradigm shift has emerged with the introduction of Vision Transformers (ViTs).
- ViTs leverage the innovative Transformer architecture, originally designed for sequential data, and apply it to image understanding.
- CNNs operate directly on pixel-level data, exploiting spatial hierarchies and local patterns.
- In contrast, ViTs treat images as sequences of patches, borrowing a page from NLP where words are treated as tokens.
- This fundamental difference in data processing coupled with the power of self-attention, enables ViTs to learn intricate patterns and relationships within images, gives ViTs a unique advantage.



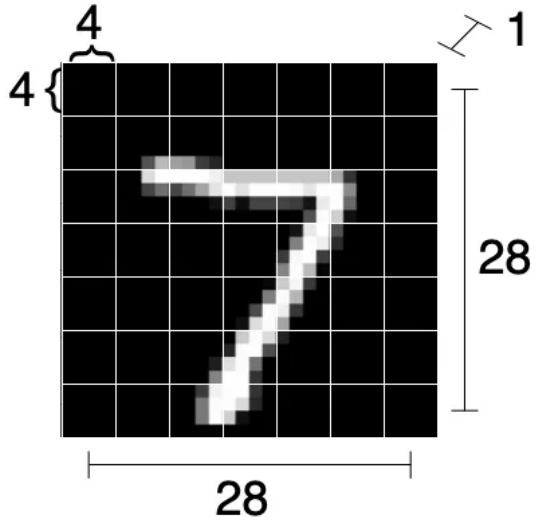
Vision Transformers



How Vision Transformers work?

Step 1: Patchifying and the linear mapping

- The transformer encoder was developed with sequence data in mind, such as English sentences.
- However, an image is not a sequence.
- We break it into multiple sub-images and map each sub-image to a vector!



Step 2: Positional encoding

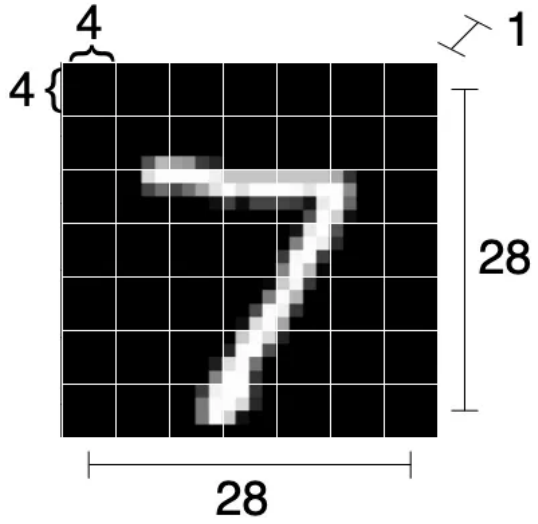
- Positional encoding allows the model to understand where each patch would be placed in the original image.
- While it is theoretically possible to learn such positional embeddings, previous work by **Vaswani et. al.** [reference] suggests that we can just add sines and cosines waves.
- In particular, positional encoding adds high-frequency values to the first dimensions and low-frequency values to the latter dimensions.

$$p_{i,j} = \begin{cases} \sin\left(\frac{i}{10000^{\frac{j}{d_{emb\_dim}}}}\right) & \text{if } j \text{ is even} \\ \cos\left(\frac{i}{10000^{\frac{j-1}{d_{emb\_dim}}}}\right) & \text{if } j \text{ is odd} \end{cases}$$

How Vision Transformers work?

Step 1: Patchifying and the linear mapping

- The transformer encoder was developed with sequence data in mind, such as English sentences.
- However, an image is not a sequence.
- We break it into multiple sub-images and map each sub-image to a vector!



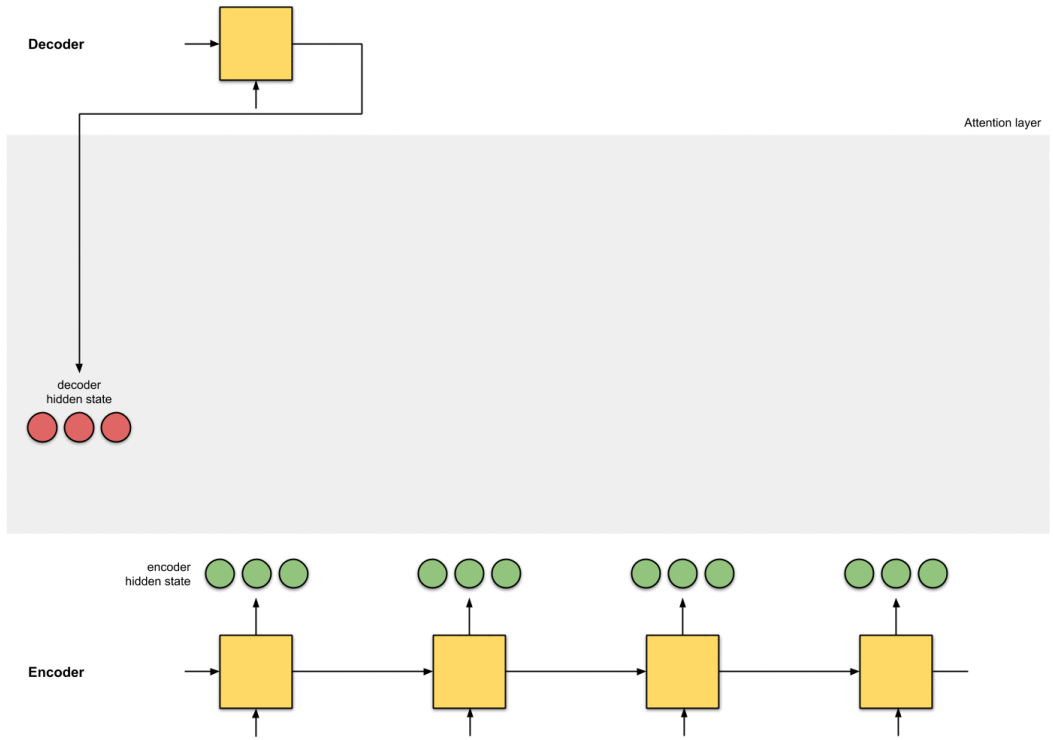
Step 2: Positional encoding

- Positional encoding allows the model to understand where each patch would be placed in the original image.
- While it is theoretically possible to learn such positional embeddings, previous work by **Vaswani et. al.** [reference] suggests that we can just add sines and cosines waves.
- In particular, positional encoding adds high-frequency values to the first dimensions and low-frequency values to the latter dimensions.

$$p_{i,j} = \begin{cases} \sin\left(\frac{i}{10000^{\frac{j}{d_{emb\_dim}}}}\right) & \text{if } j \text{ is even} \\ \cos\left(\frac{i}{10000^{\frac{j-1}{d_{emb\_dim}}}}\right) & \text{if } j \text{ is odd} \end{cases}$$

Attention Mechanism

- Attention mechanisms are designed to enhance the ability of neural networks to **focus on the most relevant parts of the input data**.
- In the context of convolutional networks, attention mechanisms can be used to improve performance by allowing the network to **emphasize important features while suppressing less relevant ones.**



Attention Mechanisms

Attention Mechanism	Purpose	Application
Self-Attention	Weighs importance of elements within the same sequence.	NLP (e.g., Transformer, BERT)
Multi-Head Attention	Applies multiple self-attentions in parallel.	Transformer architecture
Cross-Attention	Aligns and integrates information from different sequences.	Image captioning, translation
Global Context Attention	Captures long-range dependencies with global information.	Summarization, long documents
Non-Local Attention	Computes interactions between all positions in the input.	Video processing, computer vision
Squeeze-and-Excitation (SE)	Enhances feature representations by recalibrating channels.	Image classification (e.g., SENet)
Attention on Attention (AoA)	Applies attention to attention weights for further refinement.	Advanced NLP tasks

Cross-Attention Mechanism

Basic Equation: Cross-Attention(Q, K, V) = Softmax($QK^T/\sqrt{d\_k}$)V

Where:

- Q (Query): Represents the input sequence you want to attend from
- K (Key): Represents the input sequence you want to attend to
- V (Value): The actual values to be weighted
- d_k: Dimension of the key vectors (scaling factor)

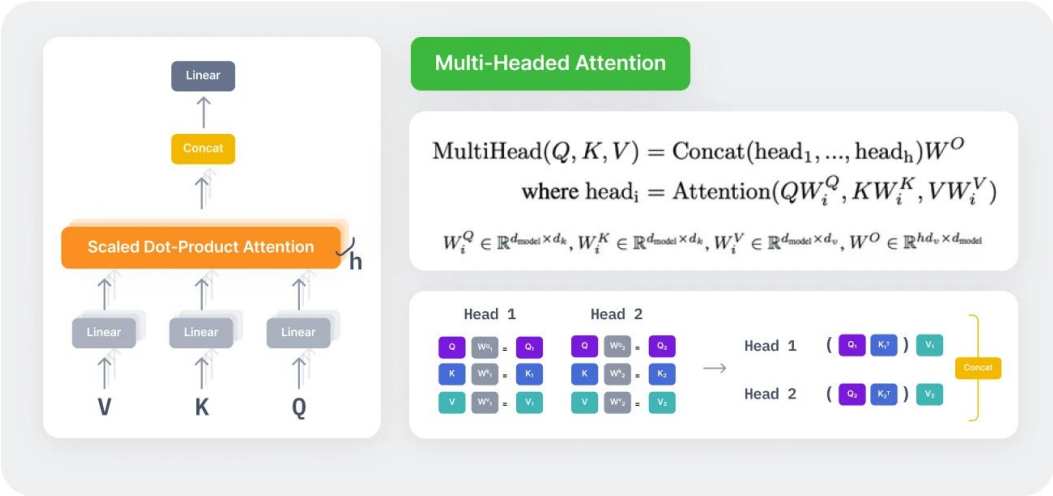
Key Properties:

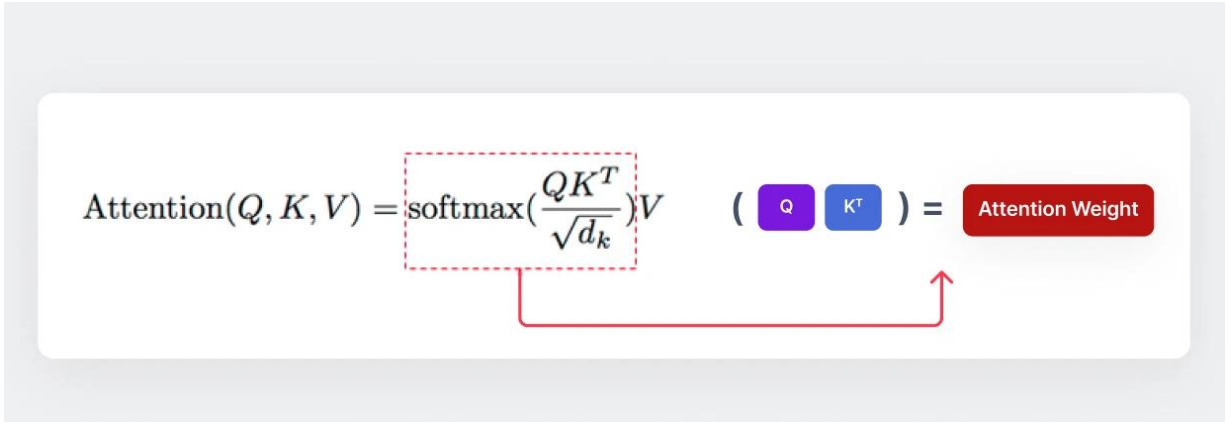
- Allows interaction between two different feature spaces
- Enables dynamic weighting of information
- Computationally complex: $O(n^2)$ complexity

Encoder

How Vision Transformers work? – Attention Mechanism

- The attention mechanism used in the Transformer uses three variables: **Q** (**Query**), **K**(**Key**), and **V** (**Value**).
- It calculates the attention weight of a Query token (token: something like a word) and a Key token and multiplies the Value associated with each Key.
- In short, it calculates the association (attention weight) between the Query token and the Key token and multiplies the Value associated with each Key.





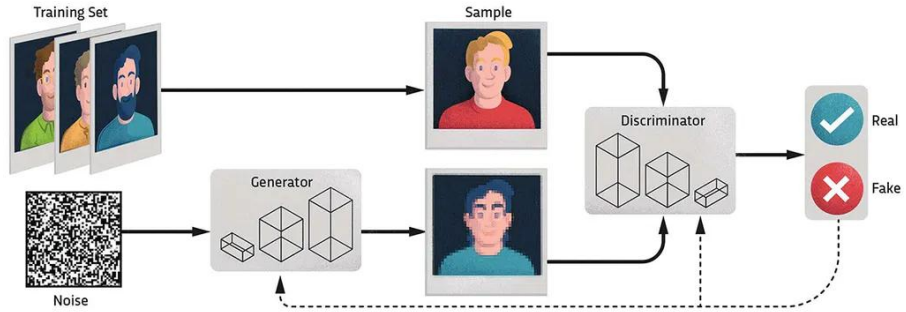
How Vision Transformers work?

Image Synthesis

Generative Adversarial Networks (GANs)

Overview

- GANs consist of two neural networks: a **generator** and a **discriminator**.
- The generator creates fake data that mimic real data.
- The discriminator evaluates the authenticity of the data (whether it's real or generated).
- The two networks are trained simultaneously in a zero-sum game until the generator produces realistic data that the discriminator cannot distinguish from real data.



Strengths:

- High-quality image generation.
- Versatility in various applications such as image-to-image translation, super-resolution, and style transfer.
- Can generate sharp and high-resolution images.

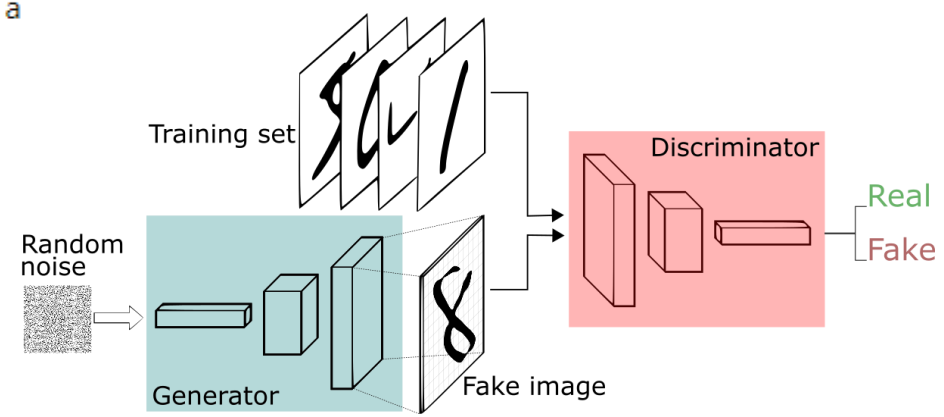
Weaknesses:

- Training can be unstable and requires careful tuning.
- Mode collapse, where the generator produces limited variations of data.
- Requires large amounts of data for effective training.

Generative Adversarial Networks (GANs) – Mathematical concept

Formulation:

- **Generator**: $G(z;\theta_G)$, where z is a noise vector sampled from a prior distribution $p_z(z)$ (e.g., a Gaussian distribution).
- **Discriminator**: $D(x;\theta_D)$, where x is either a real data point or a generated data point.



Loss Functions:

The objective function for GANs is given by:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

- The discriminator D tries to maximize the probability of correctly classifying real data x and generated data $G(z)$.
- The generator G tries to minimize the probability that the discriminator D correctly identifies $G(z)$ as fake.

Image Synthesis

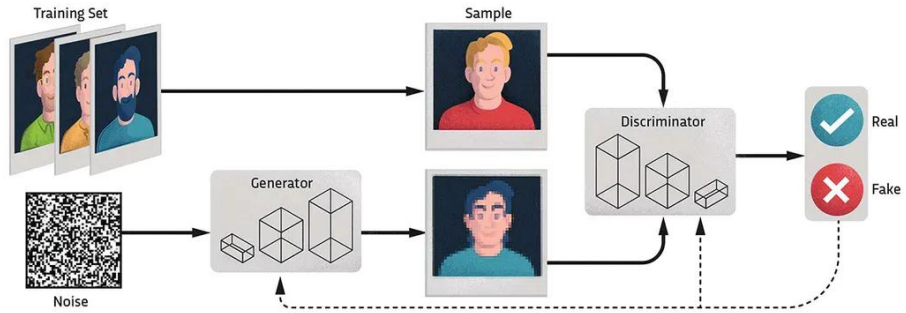
Fusion Diffusion Models (FDMs)

Overview:

- Fusion Diffusion Models, like Denoising Diffusion Probabilistic Models (DDPMs), generate data by simulating the gradual process of data corruption and subsequent denoising.
- These models start from noise and iteratively refine it through a diffusion process guided by a learned noise model.

Strengths:

- Stable and reliable training process.
- Can generate diverse samples without mode collapse.
- Strong theoretical foundation in probabilistic modeling.



Weaknesses:

- Generally slower in inference compared to GANs due to iterative refinement.
- Requires significant computational resources for training.
- Produces results that might be less sharp compared to GANs.

Fusion Diffusion Models (e.g., Denoising Diffusion Probabilistic Models, DDPMs)–

Mathematical concept

Formulation:

- **Forward Process**: A predefined Markov chain that adds Gaussian noise to data x_0 over T steps.

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)\mathbf{I})$$

where α_t are variance schedule parameters.

- **Reverse Process**: A learned Markov chain that denoises the data step by step.

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Objective Function:

The model is trained to maximize the likelihood of the data by minimizing the variational bound on the negative log-likelihood:

$$L = \mathbb{E}_{q(x_{0:T})}\left[-\log p_\theta(x_0|x_1) + \sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))\right]$$

For DDPMs, simplifying assumptions lead to a simpler objective:

$$L_{\text{simple}} = \sum_{t=1}^{T} \mathbb{E}_{x_0,\epsilon}\left[||\epsilon - \epsilon_\theta(x_t, t)||^2\right]$$

where ϵ is the noise added to x_0 to get x_t, and ϵ_θ is the model's estimate of the noise.

Comparative Analysis

Feature	GANs	Fusion Diffusion Models (e.g., DDPMs)
Architecture	Generator G and discriminator D	Noise model with iterative denoising ϵ_θ
Forward Process	N/A	Adds noise step-by-step
Reverse Process	N/A	Denoises step-by-step
Objective	Minimax game: $\min_G \max_D V(D,G)$	Maximizing likelihood by minimizing a variational bound
Loss Function	$\mathcal{L}_D$, $\mathcal{L}_G$ as described above	$\sum_{t=1}^{T} \mathbb{E}_{x_0,\epsilon}\left[\|\epsilon - \epsilon_\theta(x_t,t)\|^2\right]$
Training Stability	Can be unstable, requires careful tuning	Generally stable
Inference Speed	Fast (single pass)	Slow (iterative process)
Sample Quality	High-quality, sharp images	Diverse and complex samples, may be less sharp
Applications	Image/video generation, data augmentation	Image/video generation, speech synthesis, drug discovery

Sample Questions

- How would you implement a neural network for binary classification using Mean Squared Error (MSE) loss function? Describe the steps involved in feed-forward and back-propagation processes.

- Analyze the differences between Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs) in terms of their architecture and data processing methods. How do these differences impact their performance in image recognition tasks?

- Evaluate the strengths and weaknesses of Generative Adversarial Networks (GANs) and Fusion Diffusion Models (FDMs) for image synthesis. Which model would you recommend for generating high-quality images and why?

Question and Answer Session