Few-shot Learning

Learning with Limited Supervision

Problem

- Deep learning models $\Longrightarrow$ heavy reliance on labeled data during training
- Not directly suited to learn from few samples
- Regularization techniques reduce overfitting in low data regimes, but do not solve the problem

Learning with Limited Supervision

Problem

- Deep learning models $\Longrightarrow$ heavy reliance on labeled data during training
- Not directly suited to learn from few samples
- Regularization techniques reduce overfitting in low data regimes, but do not solve the problem

Solution

- Train models capable of rapidly generalizing to new tasks with only a few samples
- Enable models to perform under practical scenarios where data annotation is infeasible or new classes are dynamically included with time

Credit: Chen et al, A Closer Look at Few-Shot Classification, ICLR 2019

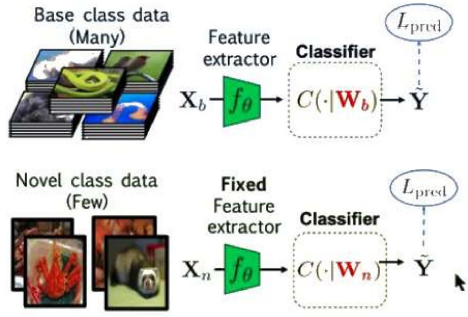
Learning with Limited Supervision

Problem

- Deep learning models $\Longrightarrow$ heavy reliance on labeled data during training
- Not directly suited to learn from few samples
- Regularization techniques reduce overfitting in low data regimes, but do not solve the problem

Solution

- Train models capable of rapidly generalizing to new tasks with only a few samples
- Enable models to perform under practical scenarios where data annotation is infeasible or new classes are dynamically included with time



Credit: Chen et al, A Closer Look at Few-Shot Classification, ICLR 2019

Problem Setting

Let x denote an image/feature (produced by a pre-trained network), y denote label

Few-Shot Learning (FSL)

- Training data $D_{train} = (x_i, y_i)_{i=1}^{I}$, where few training samples for certain classes
- Specifically, **N-way-K-shot FSL** problem: D_{train} contains only few examples, K, from N of the overall number of classes (other classes called **base classes**)

Problem Setting

Let x denote an image/feature (produced by a pre-trained network), y denote label

Few-Shot Learning (FSL)

- Training data $D_{train} = (x_i, y_i)_{i=1}^{I}$, where few training samples for certain classes
- Specifically, **N-way-K-shot FSL** problem: D_{train} contains only few examples, K, from N of the overall number of classes (other classes called **base classes**)

Zero-Shot Learning (ZSL)

- Training data $S = \{(x, y, a(y)) | x \in X^s, y \in Y^s, a(y) \in A\}$, where X^s is set of image/features from seen classes, Y^s is set of seen class labels, $a(y)$ is semantic embedding for class y
- Test set $U = \{(x, y, a(y)) | x \in X^u, y \in Y^u, a(y) \in A\}$, where X^u is set of unseen class image/features, Y^u is set of unseen class labels, $Y^u \cap Y^s = \emptyset$

Problem Setting

Based on classes that a model sees in test phase, FSL/ZSL problem generally categorized into two settings:

Conventional FSL/ZSL:

- Goal is to learn a classifier $f : x \rightarrow Y^u$
- Image/feature x to be recognized at test time belongs only to unseen/few-shot classes

Problem Setting

Based on classes that a model sees in test phase, FSL/ZSL problem generally categorized into two settings:

Conventional FSL/ZSL:

- Goal is to learn a classifier $f : x \rightarrow Y^u$
- Image/feature x to be recognized at test time belongs only to unseen/few-shot classes

Generalized FSL/ZSL:

- Goal is to learn a classifier $f : x \rightarrow Y^u \cup Y^u$
- Image/feature x to be recognized at test time may belong to seen/base or unseen/few-shot classes
- Practically more useful and challenging than conventional setting, since assumption that images at test time come only from unseen/few-shot classes need not hold

Recall: Supervised Learning

Empirical Risk Minimization

Let $p(x, y)$ be ground-truth joint probability distribution of input x and output y

- Given hypothesis h, we want to minimize its expected risk R, loss measured w.r.t. $p(x, y)$, i.e.

$$R(h) = \int l(h(x), y) d(p(x, y)) = E[l(h(x), y)]$$

Recall: Supervised Learning

Empirical Risk Minimization

Let $p(x, y)$ be ground-truth joint probability distribution of input x and output y

- Given hypothesis h, we want to minimize its expected risk R, loss measured w.r.t. $p(x, y)$, i.e.

$$R(h) = \int l(h(x), y) d(p(x, y)) = E[l(h(x), y)]$$

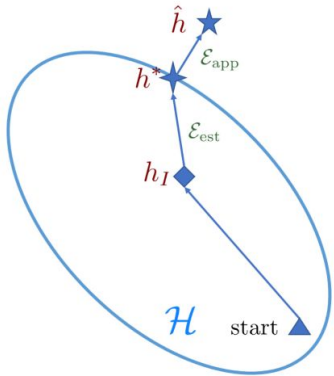
- As $p(x, y)$ is unknown, empirical risk $R_I(h)$ is used as proxy for $R(h)$, leading to empirical risk minimization:

$$R_I(h) = \frac{1}{I} \sum_{i=1}^{I} l(h(x_i, y_i))$$

Recall: Supervised Learning

Let $h \in H$ be a hypothesis from x to y in hypothesis space defined by H

- $\hat{h} = \arg\min_h R(h)$ be function that minimizes expected risk;
  $h^* = \arg\min_{h \in H} R(h)$ be function in H that minimizes expected risk
  $h_I = \arg\min_{h \in H} R_I(h)$ be function in H that minimizes empirical risk



Recall: Supervised Learning

Let $h \in H$ be a hypothesis from x to y in hypothesis space defined by H

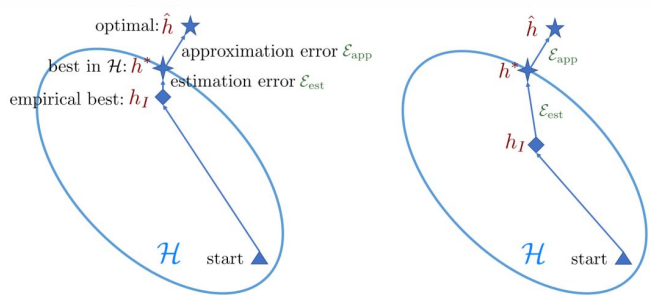
- $\hat{h} = \arg\min_h R(h)$ be function that minimizes expected risk;
  $h^* = \arg\min_{h \in H} R(h)$ be function in H that minimizes expected risk
  $h_I = \arg\min_{h \in H} R_I(h)$ be function in H that minimizes empirical risk
- **Error Decomposition:**

$$E[R(h_I) - R(\hat{h})] = \underbrace{E[R(h^*) - R(\hat{h})]}_{\mathcal{E}_{app}(H)} + \underbrace{E[R(h_I) - R(h^*)]}_{\mathcal{E}_{est}(H,I)}$$

$\mathcal{E}_{app}$ (**approximation error**) measures how closely functions in H can approximate optimal hypothesis $\hat{h}$
$\mathcal{E}_{est}$ (**estimation error**) measures effect of minimizing empirical risk $R_I(h)$ instead of expected risk $R(h)$ within H

Problems with Few/Zero-shot Setting

- $\mathcal{E}_{est} \implies$ can be reduced with large training dataset
- Sufficient labeled train data with (i.e. large I) $\implies$ Empirical risk minimizer $R(h_I)$ gives good approximation to best possible $R(h^*)$

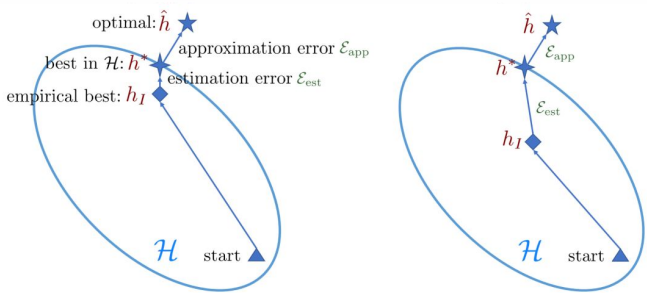


(a) Large I (b) Small I

Fig. 1. Comparison of learning with sufficient and few training samples.

Problems with Few/Zero-shot Setting

- $\mathcal{E}_{est} \implies$ can be reduced with large training dataset
- Sufficient labeled train data with (i.e. large I) $\implies$ Empirical risk minimizer $R(h_I)$ gives good approximation to best possible $R(h^*)$



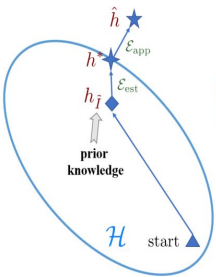
(a) Large I (b) Small I

Fig. 1. Comparison of learning with sufficient and few training samples.

Few-shot Learning:

- Number of labeled examples I is small
- $R_I(h) \implies$ far from being good approximation of expected risk $R(h)$
- Resultant empirical risk minimizer h_I overfits

Addressing Few/Zero-shot Learning

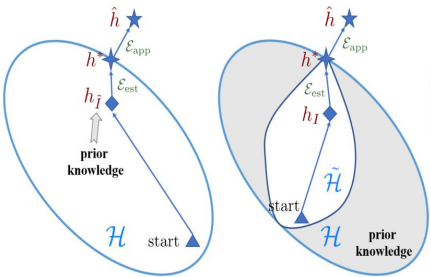


(a) Data.

Data

- Learn to augment training set $\Longrightarrow$ increase number of samples to $\tilde{I} >> I$
- More accurate empirical risk minimizer $h_{\tilde{I}}$ can be obtained

Addressing Few/Zero-shot Learning



(a) Data. (b) Model.

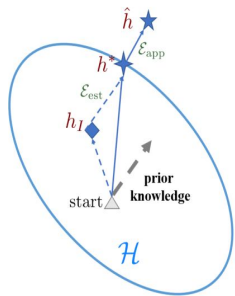
Data

- Learn to augment training set $\Longrightarrow$ increase number of samples to $\tilde{I} >> I$
- More accurate empirical risk minimizer $h_{\tilde{I}}$ can be obtained

Model

- Constrain complexity of H $\Longrightarrow$ much smaller hypothesis space $\tilde{H}$
- Then, D_{train} may be sufficient to learn a reliable h_I

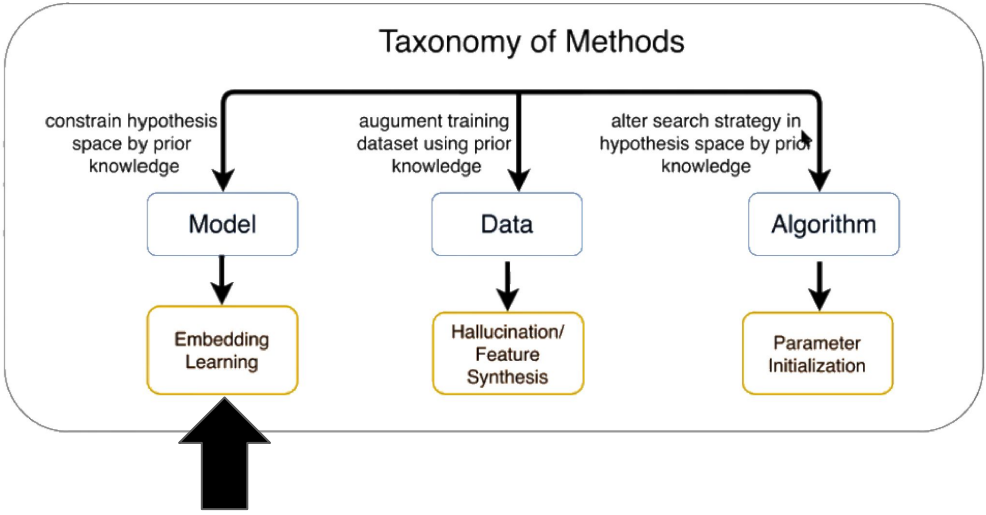
Addressing Few/Zero-shot Learning



(c) Algorithm.

Algorithm

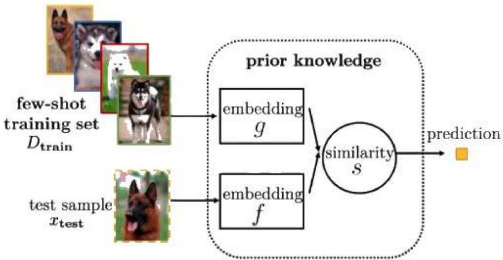
- Search for parameters θ for best hypothesis h^* in H
- Prior knowledge alters search strategy by providing a good initialization (gray triangle in Fig (c))


Taxonomy of Methods
constrain hypothesis space by prior knowledge
augument training dataset using prior knowledge
alter search strategy in hypothesis space by prior knowledge
Model
Data
Algorithm
Embedding Learning
Hallucination/ Feature Synthesis
Parameter Initialization

Embedding Learning Methods

Intuition:

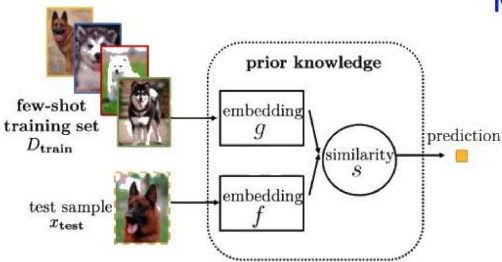
- Address few-shot learning by "learning to compare"
- If model can determine similarity of two images (and perhaps corresponding semantics of classes), it can classify unseen input in relation to labeled instance seen during training



Embedding Learning Methods

Intuition:

- Address few-shot learning by "learning to compare"
- If model can determine similarity of two images (and perhaps corresponding semantics of classes), it can classify unseen input in relation to labeled instance seen during training

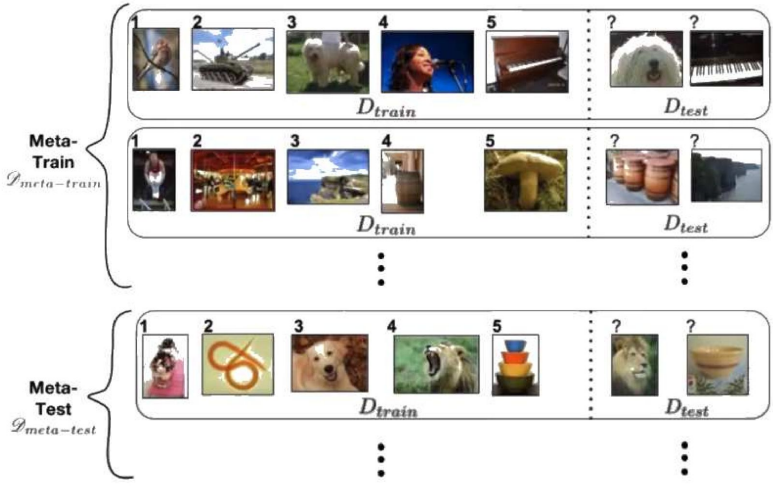


Method:

- Learn separate embedding functions for training samples D_{train} and test samples D_{test}
- Train sophisticated comparison models end-to-end via **meta-learning**
- At test time, predict based on comparing distance between x_{test} feature and training set features from each class

Problem Setup: Meta-Learning

- **N-way-K-shot:** N different classes in D_{train} with K samples per class
- $D_{meta-train} = (D_{train}, D_{test})$ set $\rightarrow$ one task/episode
- Ensure $D_{meta-train}$ and $D_{meta-test}$ have disjoint/different classes



Problem Setup: Meta-Learning

Learning Algorithm A

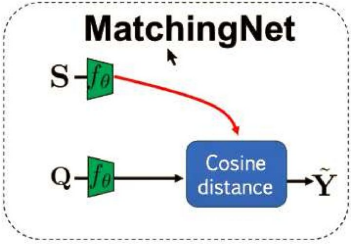
- **Input**: Training set $D_{train} = (x_i, y_i)_{i=1}^{I}$
- **Output**: Parameter θ model M (the **learner**)
- **Objective**: Good performance on $D_{test} = (x'_i, y'_i)$

Meta-Learning Algorithm

- **Input**: Meta-training set $D_{meta-train} = (D_{train}^{(n)}, D_{test}^{(n)})_{n=1}^{N}$ of tasks/episodes
- **Output**: Parameter Θ algorithm A(the **meta-learner**)
- **Objective**: Good performance on meta test set $D_{meta-test} = (D_{train}^{\prime(n)}, D_{train}^{\prime(n)})_{n=1}^{N}$

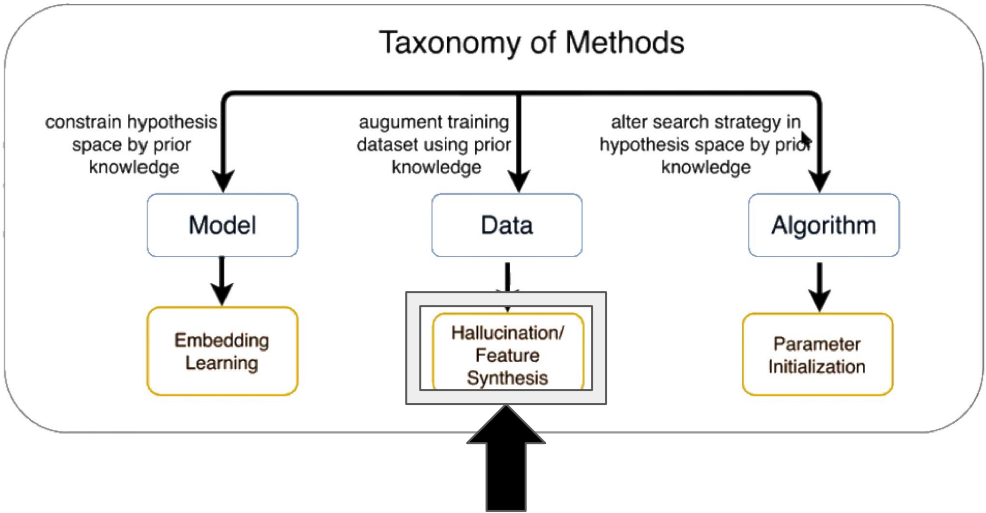
Matching Networks

- **Parametric models:** Slowly learn model parameters from training examples; Require large datasets to avoid overfitting
- **Non-parametric models:** Allow novel examples to be rapidly assimilated; Robust to **catastrophic forgetting**



Idea: Combine best of both worlds

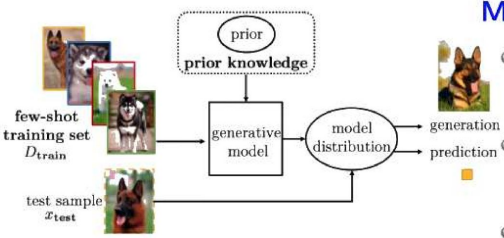
- **Training Phase:** Learn cosine similarity-based embedding models (parametric meta-learners)
- **Test Phase:** Use Nearest Neighbors (non-parametric) in learned embedding space


Taxonomy of Methods
constrain hypothesis space by prior knowledge
augument training dataset using prior knowledge
alter search strategy in hypothesis space by prior knowledge
Model
Data
Algorithm
Embedding Learning
Hallucination/ Feature Synthesis
Parameter Initialization

Hallucination/Feature Synthesis Methods

Intuition:

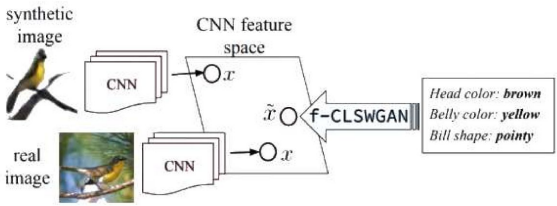
- Directly deal with data deficiency by "learning to augment"
- Learn a generative model to hallucinate new novel class data for data augmentation
- Reduce few/zero-shot problem to a standard supervised learning problem



Method:

- Learn a generator conditioned on meta-information using data in base classes
- Generate novel class features conditioned on unseen class meta-information
- Train a classifier on base class samples and generated novel class samples

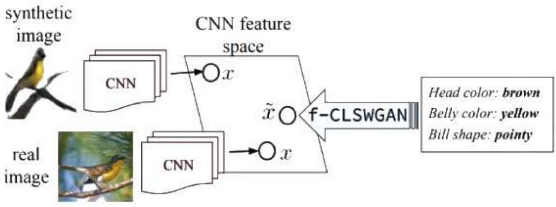
Feature Generating Networks (f-CLSWGAN)



Method:

- Given train set S of seen classes, learn a conditional generator $G : Z \times C \rightarrow X$, which takes random Gaussian noise $z \in Z \subset R^{dz}$ and class embedding $c(y) \in C$, and outputs image feature $\tilde{x} \in X$
- To ensure $\tilde{x}$ are well-suited to train a discriminative classifier, minimize classification loss over generated features $\tilde{x}$

Feature Generating Networks (f-CLSWGAN)



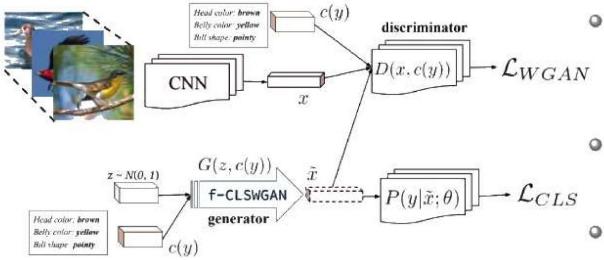
Method:

- Given train set S of seen classes, learn a conditional generator $G : Z \times C \rightarrow X$, which takes random Gaussian noise $z \in Z \subset R^{dz}$ and class embedding $c(y) \in C$, and outputs image feature $\tilde{x} \in X$
- To ensure $\tilde{x}$ are well-suited to train a discriminative classifier, minimize classification loss over generated features $\tilde{x}$

Extension to Few-shot Learning:

- For FSL, along with seen classes data set S, the training data also includes few labeled samples for each unseen class as well

Feature Generating Networks (f-CLSWGAN)



Loss Formulation:

- **GAN Loss:**

$$\mathcal{L}_{WGAN} = E[D(x, c(y))] - E[D(\tilde{x}, c(y))] - \lambda E[(||\nabla_{\hat{x}} D(\hat{x}, c(y))||_2 - 1)^2]$$

- **Classification Loss:**

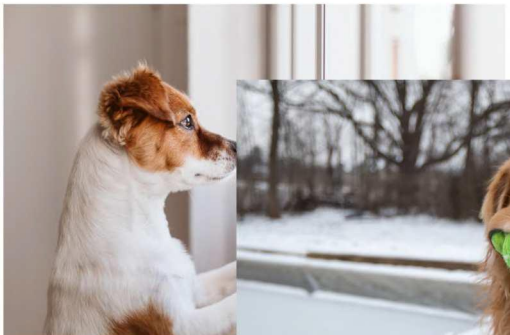
$$\mathcal{L}_{CLS} = -E_{\tilde{x} \sim p_{\tilde{x}}}[\log P(y|\tilde{x};\theta)]$$

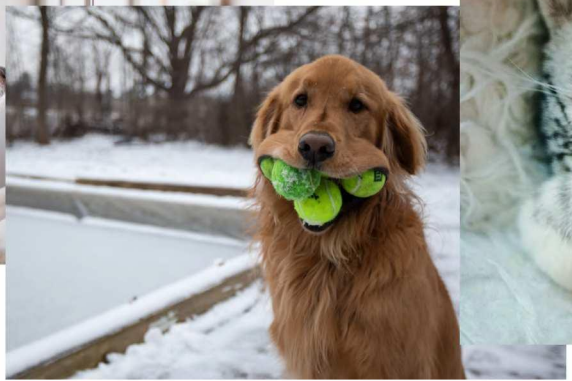
- **Final loss:**

$$L_{total} = \min_G \max_D \mathcal{L}_{WGAN} + \beta \mathcal{L}_{CLS}$$

Vision and Language

Describe these images

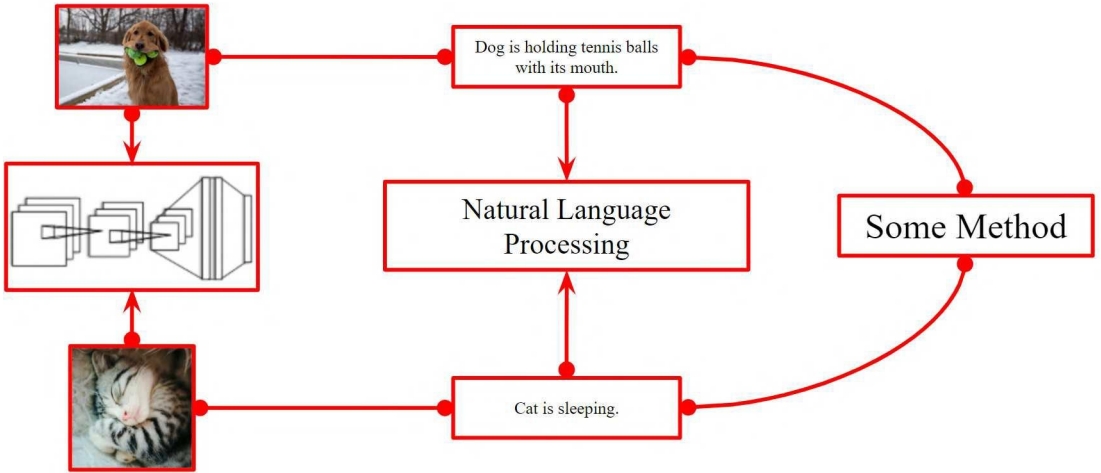




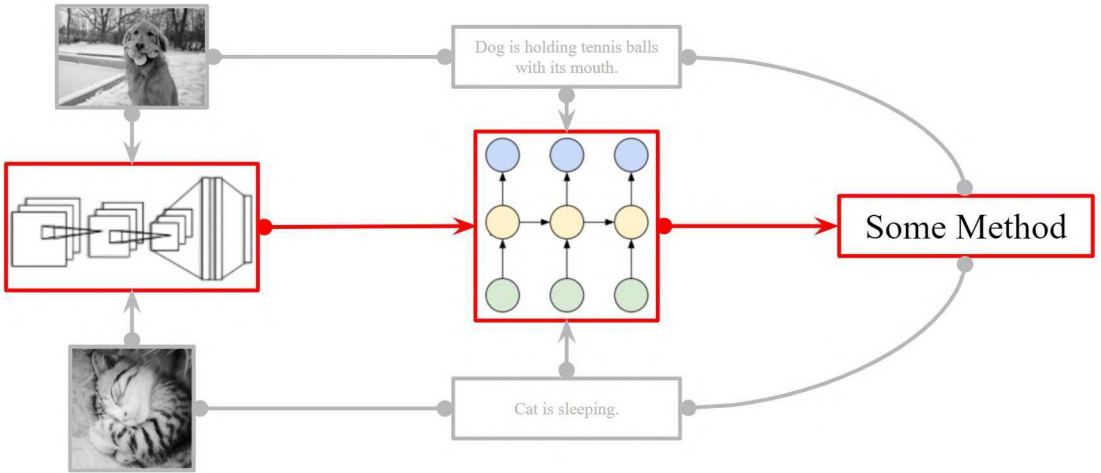
Describe these images

- How can we understand what is happening just by looking at a single image ? Can we make a computer do the same?

How to make a computer describe an image?



How to make a computer describe an image?



How to make a computer describe an image?

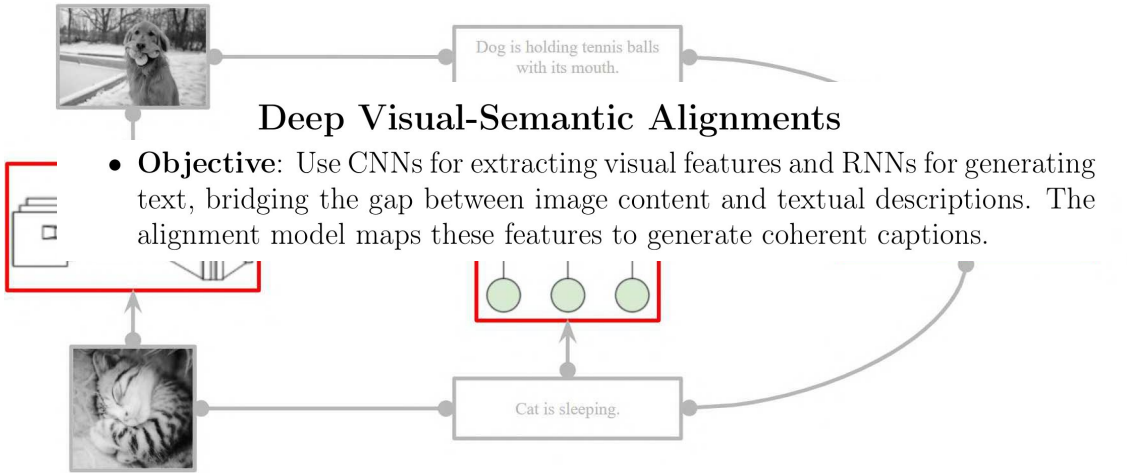
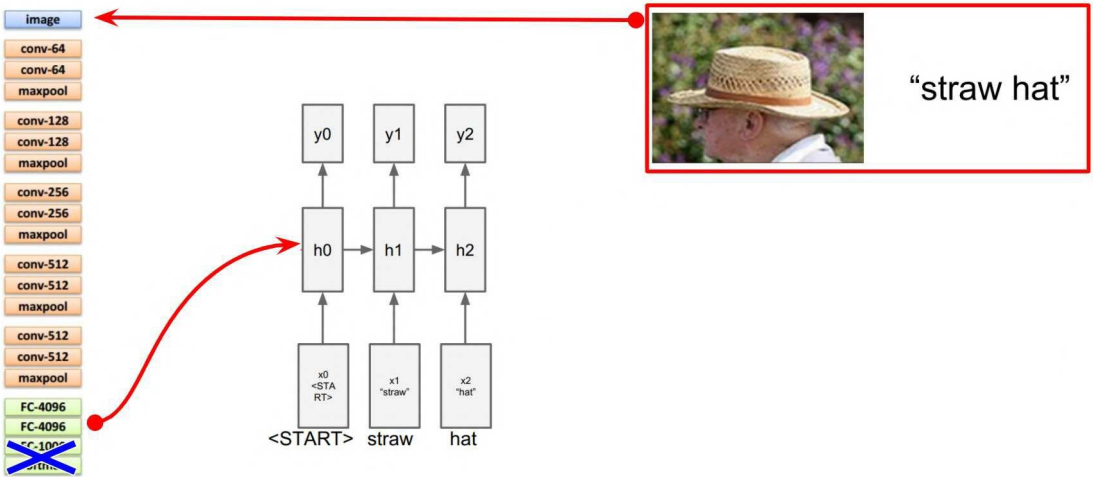
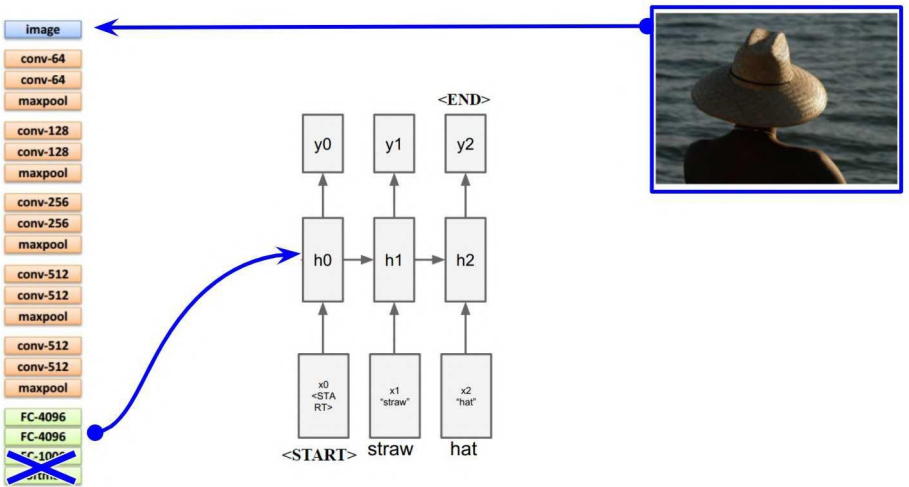


Image Captioning: Training



Credit: Karpathy et al, Deep visual-semantic alignments for generating image descriptions, CVPR 2015

Image Captioning: Inference (Test Time)



Credit: Karpathy et al, Deep visual-semantic alignments for generating image descriptions, CVPR 2015

Results

a group of people standing around a room with remotes
logprob: -9.17

a young boy is holding a baseball bat
logprob: -7.61

a cow is standing in the middle of a street
logprob: -8.84

Results: Failure Cases

Possible to understand why the method failed

a man standing next to a clock on a wall
logprob: -10.08

a young boy is holding a baseball bat
logprob: -7.65

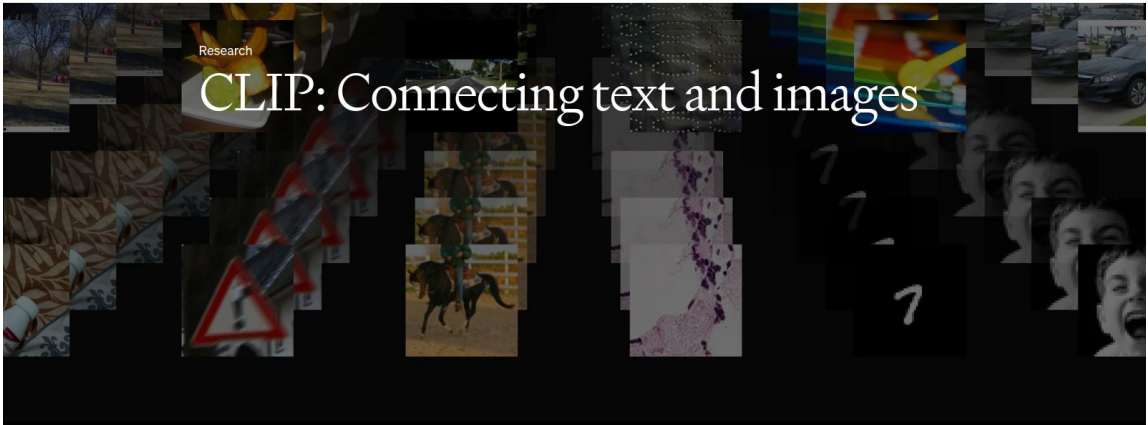
a cat is sitting on a couch with a remote control
logprob: -12.45

Results: Failure Cases

Not possible to understand why the method failed

a woman holding a teddy bear in front of a mirror
logprob: -9.65

a horse is standing in the middle of a road
logprob: -10.34



Research

CLIP: Connecting text and images

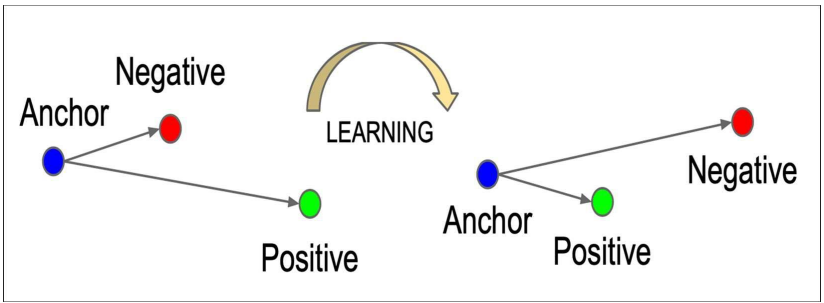
Illustra

We're introducing a neural network called CLIP which efficiently learns visual concepts from natural language supervision. CLIP can be applied to any visual classification benchmark by simply providing the names of the visual categories to be recognized, similar to the "zero-shot" capabilities of GPT-2 and GPT-3.

CLIP Model Overview

- Developed by OpenAI to learn visual concepts from natural language.
- Performs tasks across a broad spectrum without task-specific data.
- Utilizes contrastive learning, aligning text-image pairs in a shared embedding space.
- Aims to bridge the gap between visual understanding and natural language processing (NLP).

What is Contrastive Learning?



Contrastive Learning Objective - similar (image, text) pair

Input Image

$\vec{H}_i$

Image Representation

A dog lying in grass

Input Text

$\vec{H}_t$

Text Representation

$$maximize(\frac{\vec{H}_i \cdot \vec{H}_t}{\|\vec{H}_i\| \times \|\vec{H}_t\|})$$

Contrastive Learning Objective - dissimilar (image, text) pair

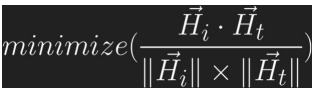
Input Image

Image Representation

A dog lying in grass

Input Text

Text Representation



How to Train CLIP?

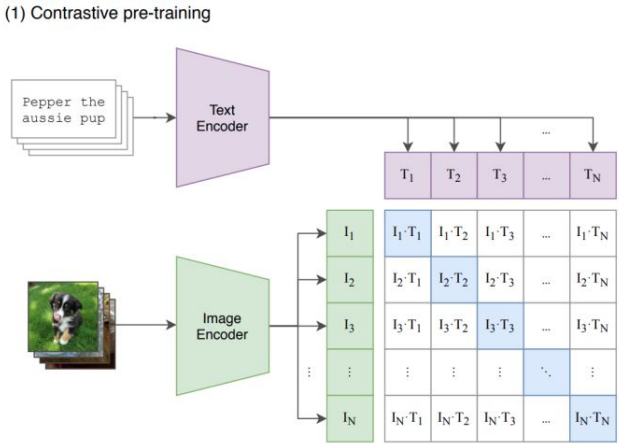
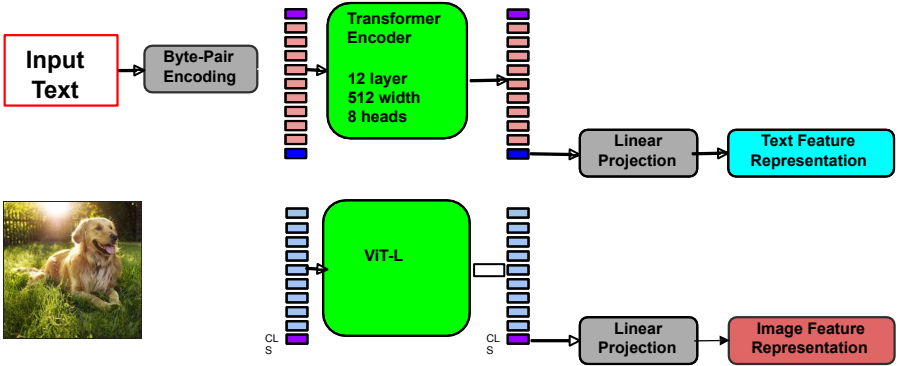


Figure: Contrastive Pre-training of language-image pairs. The text encoder is a standard transformer encoder. The extracted feature is the embedding of the CLS token. The image encoder is either a ResNet-50 or a Vision Transformer (ViT).

Embedding Functions Explained

- Embeddings transform raw data (images/text) into a high-dimensional space, capturing semantic meaning.
- Image Function: $E_I = f_{\text{image}}(I)$, where f_{image} is a deep network (e.g., Vision Transformer) extracting visual features.
- Text Function: $E_T = f_{\text{text}}(T)$, utilizing Transformer architecture to encode textual information.
- This shared space is where CLIP evaluates the congruence of visual-textual content, crucial for its versatile application across different tasks.

CLIP Architecture



* Authors also tested many other ResNet variants, but found this ViT to perform the best

How to Train CLIP?

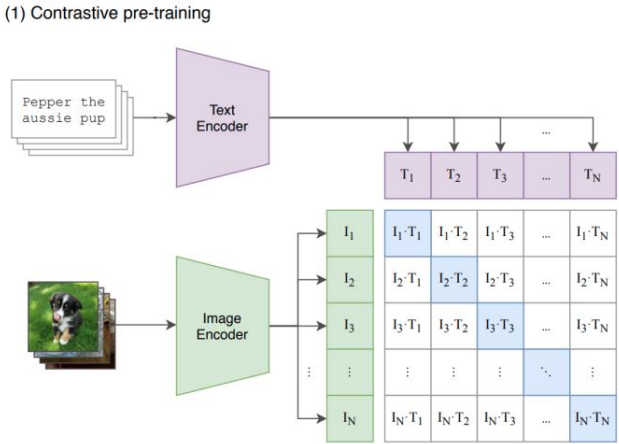
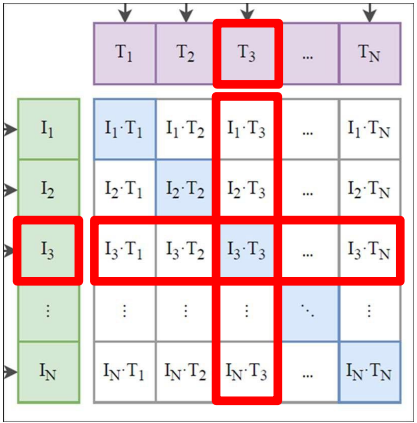


Figure: Contrastive Pre-training of language-image pairs. The text encoder is a standard transformer encoder. The extracted feature is the embedding of the CLS token. The image encoder is either a ResNet-50 or a Vision Transformer (ViT).

Computing Loss



m_i = one-hot encoded label vector for the i-th image sample

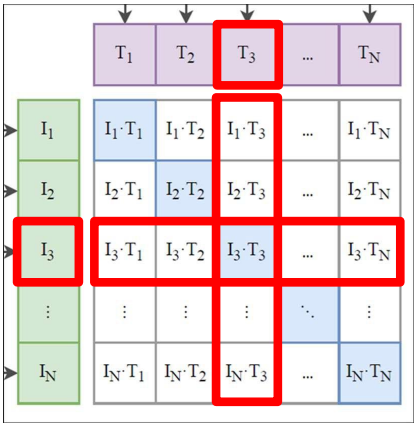
y_i^m = cosine similarities vector for i-th image sample

t_i =one-hot encoded label for the i-th text sample

y_i^t = cosine similarities vector for i-th text sample

ϕ = Cross Entropy: $C(P) = -\Sigma_i P(i) \log Q(i)$

Computing Loss



m_i = one-hot encoded label vector for the i-th image sample

y_i^m = cosine similarities vector for i-th image sample

t_i =one-hot encoded label for the i-th text sample

y_i^t = cosine similarities vector for i-th text sample

ϕ = Cross Entropy: C(P) = - Σ_iP(i) log Q(i)

$$\mathcal{L}_m = \frac{\sum_{i=1}^{N} \phi(y_i^m, m_i)}{N}$$

$$\mathcal{L}_t = \frac{\sum_{i=1}^{N} \phi(y_i^t, t_i)}{N}$$

$$\mathcal{L} = \frac{\mathcal{L}_m + \mathcal{L}_t}{2}$$

Some CLIP details

Training

- Trained on 400M image-text pairs from the internet
- Batch size of 32,768
- 32 epochs over the dataset
- Cosine learning rate decay

Architecture

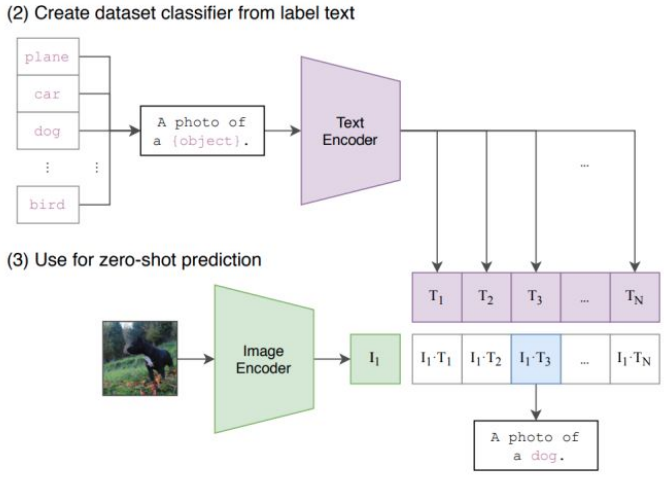
- ResNet-based or ViT-based image encoder
- Transformer-based text encoder

Testing

- Linear Probe

- Zero-shot Prediction

How to Use CLIP for Classification?



Zero-Shot Classification Mechanics

- Harnesses pre-trained embeddings to classify images into unseen categories.
- Decision rule:

$$C = \arg\max_{T} \text{sim}(I, T)$$

, leveraging similarity scores to determine the closest text label for an image.

How Well Does CLIP Do on Zero-shot Classification?

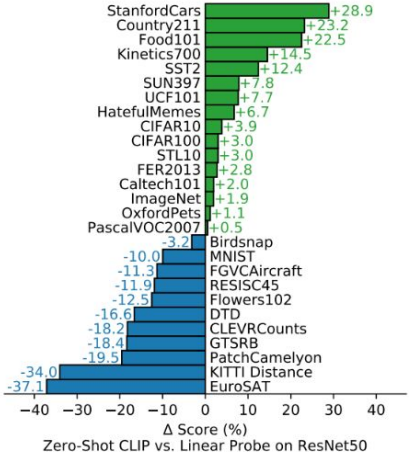


Figure: Zero-shot CLIP is competitive with a fully supervised baseline. Across a 27 dataset eval suite, a zero-shot CLIP classifier outperforms a fully supervised linear classifier fitted on ResNet-50 features on 16 datasets, including ImageNet.

Zero-shot CLIP