Hands-On Few-Shot Learning

Isha Chaturvedi

Session Outline

- Lesson 1: Fundamentals of Few Shot Learning
- Lesson 2: Siamese Neural Networks, Prototypical Networks & different loss functions
- Lesson 3: Implementation of Siamese Neural Networks with Triplet Loss on Imagery Data

Lesson 1: Fundamentals of Few Shot Learning (FSL)

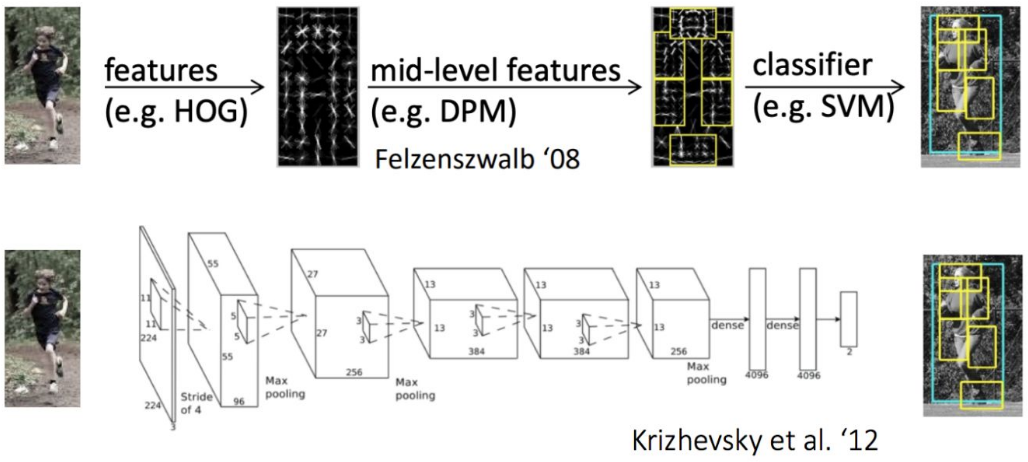
- Presentation:
  - Basic concepts of FSL, meta-learning framework (support query framework) using a fruit imagery example.
  - Difference between supervised learning and FSL, including covering the differences between few-shot, one-shot, and zero-shot learning, and their respective applications.
- Discussion: Open discussion on the practical use cases of FSL, on imagery/NLU dataset, and multi-modal dataset by showing examples.
- Q&A

Break (5 minutes)

Traditional deep learning allows us to learn unstructured features using large labeled training set

Standard computer vision:
hand-designed features

Modern computer vision:
end-to-end training



Deep learning allows us to handle *unstructured inputs* (pixels, language, sensor readings, etc.) without hand-engineering features, with less domain knowledge

Source: https://cs330.stanford.edu/

Supervised deep learning run into issues when you don’t have large dataset

Large, diverse data (+ large models) $\xrightarrow{\text{deep learning}}$ Broad generalization

Russakovsky et al. ‘14

Wu et al. ‘16

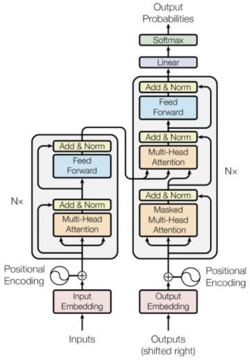


Figure 1: The Transformer - model architecture.

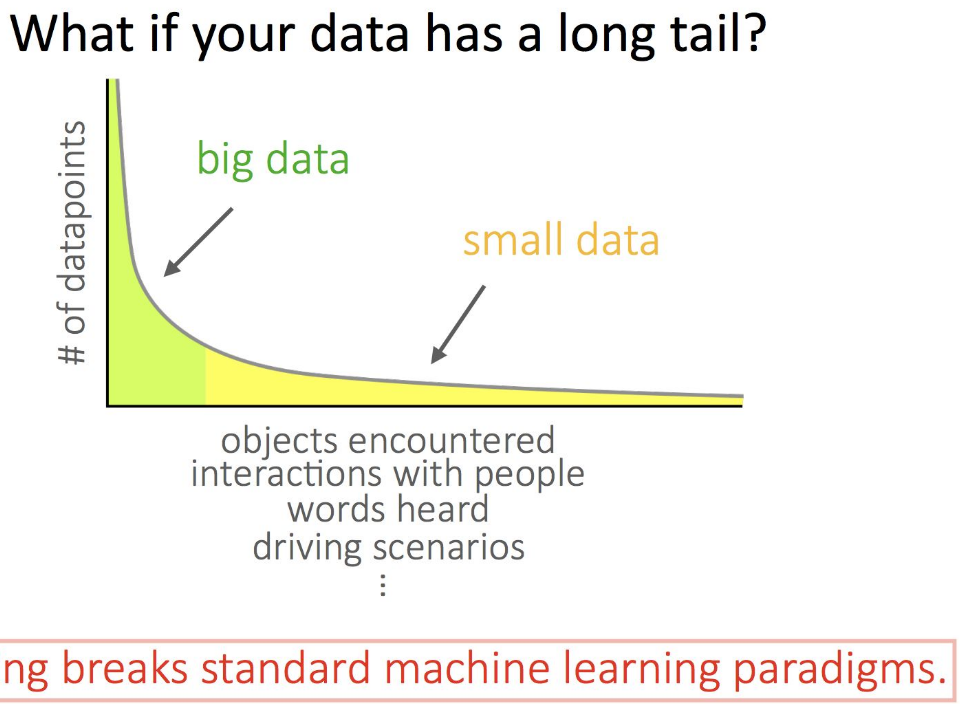
Vaswani et al. ‘18

What if you don’t have a large dataset?

Impractical to learn from scratch for each disease, each robot, each person, each language, **each task**

Source: https://cs330.stanford.edu/

A long-tailed data doesn’t fit into standard machine learning paradigm



Source: https://cs330.stanford.edu/

We can solve such tasks by leveraging previous tasks

training data

Braque Cezanne

test datapoint

By Braque or Cezanne?

Source: https://cs330.stanford.edu/

Given abundant training examples for the base classes, few-shot learning algorithms aim to learn to recognizing novel classes with a limited amount of labeled examples.

Let’s play a game where we are given 3 cards with pictures of 3 fruits

Watermelon Pineapple Grapes

Figure 1: The three card images comprising the support set

Figure 2: The card image comprising the query set

The meta-learning problem (learning to learn):

Given data on previous tasks, learn a new task more quickly and/or more proficiently.

Source: https://cs330.stanford.edu/

Paradigm on N-way K-shot problem

- N way: We are given N classes. In above fruit example, N = 3. N classes comprises the support set.
- K shot: The number of K labeled samples per shot. In above example, K = 1.
- Support set: The N*K samples we can use for learning.
- Query set: Target samples. Number can vary. In above example, Q = 1 (1 query image).
- The main task is to classify the images in the query set, using the N classes, given the N*K images in the support set. Here when K = 1, we call it **one-shot classification**.
- Task: Support + Query sets (used in different ways by different methods).
- Training: random tasks can also be in a setup of support and query (matching networks) or for learning similarity function (siamese network).
- Test: New unseen task, given the support labels, predicts the labels in query set.

The difference between few-shot, one-shot and zero-shot learning is the number of labeled samples per class

Few-Shot Learning

Today's Focus

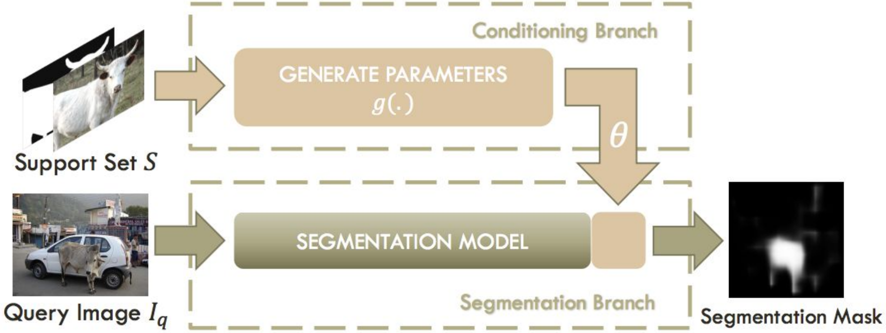
- There are different approaches to Few-Shot Learning:
    - Prior knowledge about similarity (distance based metric methods)
    - Prior knowledge about learning: to constrain the learning algorithm to choose parameters that generalize well from a few examples (e.g. MAML*)
    - Prior knowledge of data: exploit prior knowledge about the structure and variability of the data, which enables constructing viable models from a few examples.(e.g. Neural Statistician)

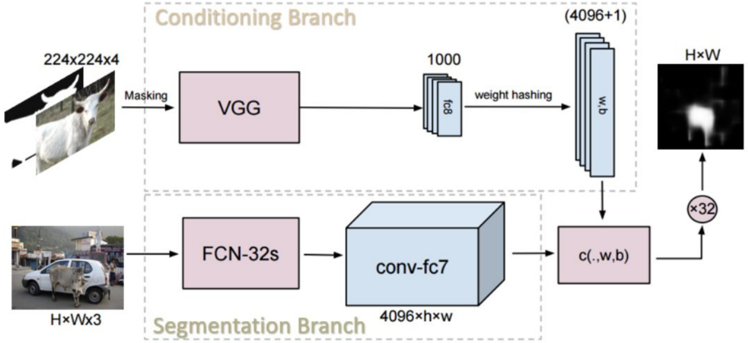
*Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks
Source: What is Few-Shot Learning? Methods & Applications

The difference between few-shot, one-shot and zero-shot learning is the number of labeled samples per class

One-Shot Learning

- Application: One-Shot Learning for Semantic Segmentation





Source: One-Shot Learning for Semantic Segmentation, What is Few-Shot Learning? Methods & Applications

The difference between few-shot, one-shot and zero-shot learning is the number of labeled samples per class

Zero-Shot Learning

- Zero-Shot Learning is the ability to detect classes that the model has never seen during training.
- SOTA e.g: Prompt based Zero-Shot Learning*

- Application: Build a model that goes from images -> attributes.

*Language Models are Few-Shot Learners
Source: Prompt-based learning, What is Few-Shot Learning? Methods & Applications

Group Discussion

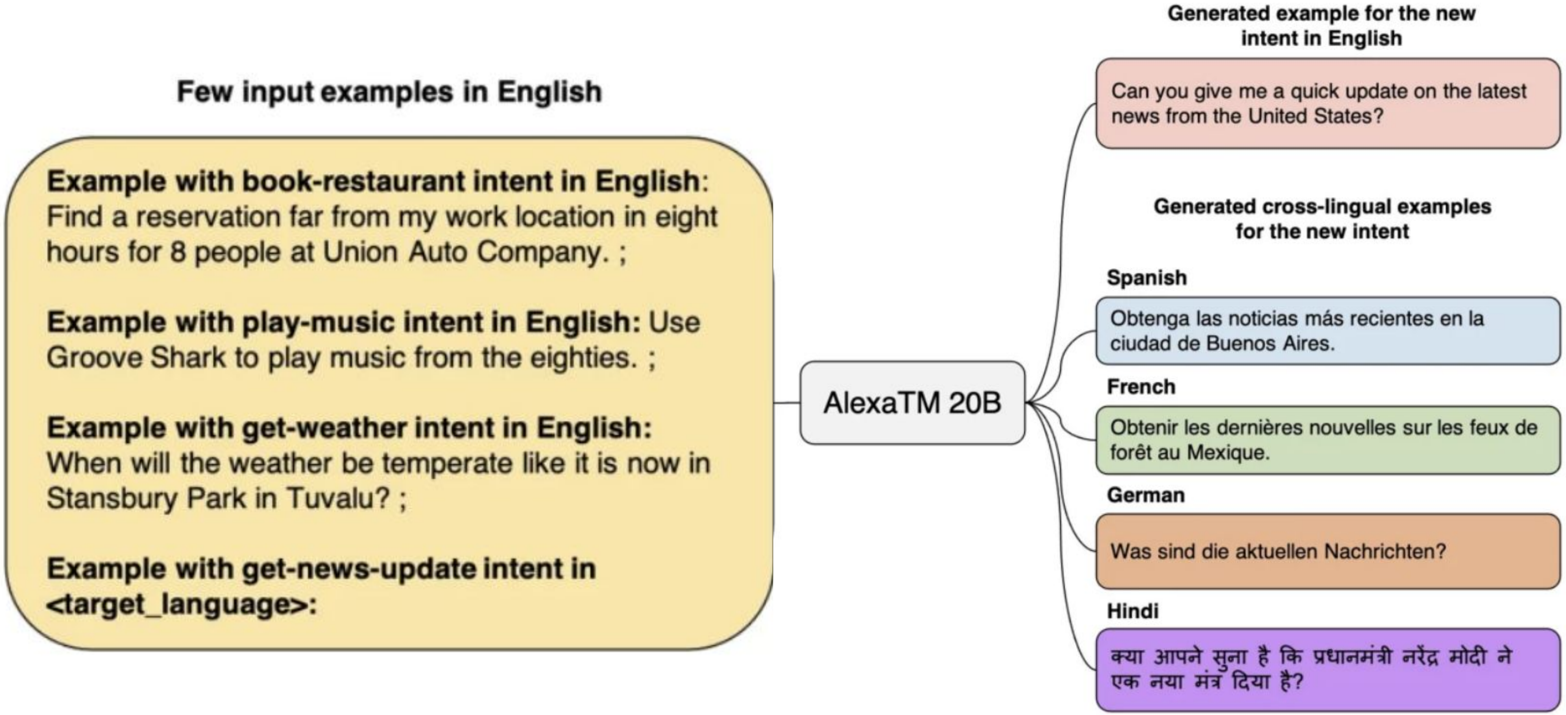
Practical use cases of FSL on imagery/NLU dataset, and multi-modal dataset

Few-Shot Learning has various applications in the most popular fields

Computer Vision	Natural Language Processing	Robotics	Audio Processing
Character & Object Recognition, Image Classification	Translation, Sentence Completion, Parsing	Visual Navigation	Voice Cloning from few audio samples of the user
Other Image Applications: image retrieval, object tracking, specific object counting in images, gesture recognition, part labeling, image view generation, image captioning	Sentence Classification, Topic Discovery, Word Similarity, Multi-Label Text Classification	Learning a movement by imitating a single demonstration, Learning manipulation of actions from few demonstrations	Voice Conversion from one user to another, Voice Conversion across different languages
Video Applications: video classification, motion prediction, action localization			

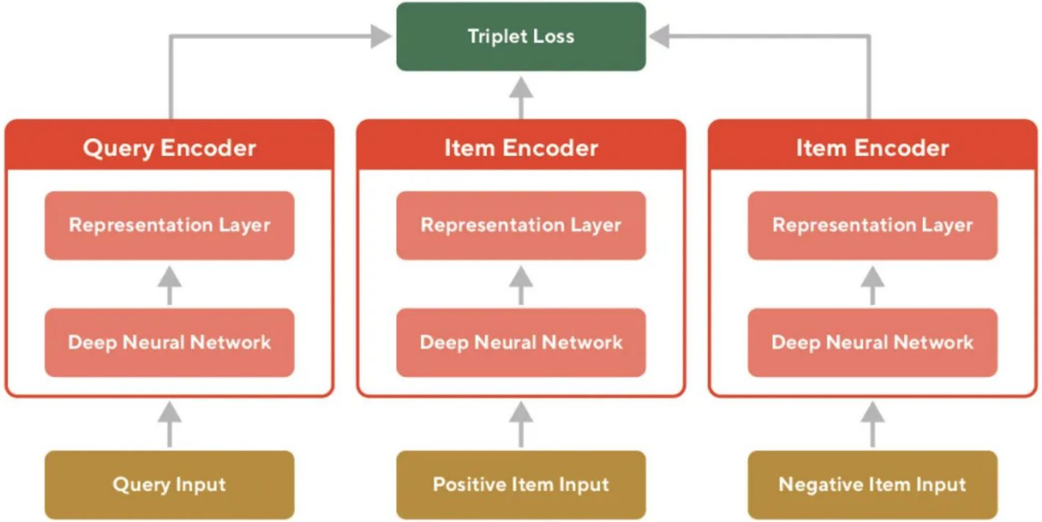
Source: Few-shot Learning Explained: Examples, Applications, Research

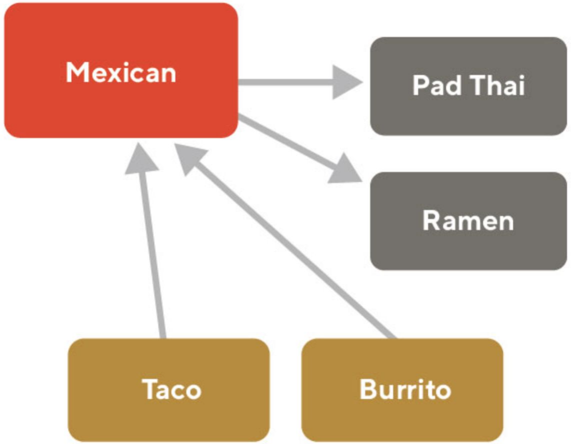
Amazon newly released AlexaTM 20B is a multilingual language super model capable of few-shot learning



Source: AlexaTM 20B is Amazon's New Language Super Model Which is Also Capable of Few-Shot Learning | by Jesus Rodriguez

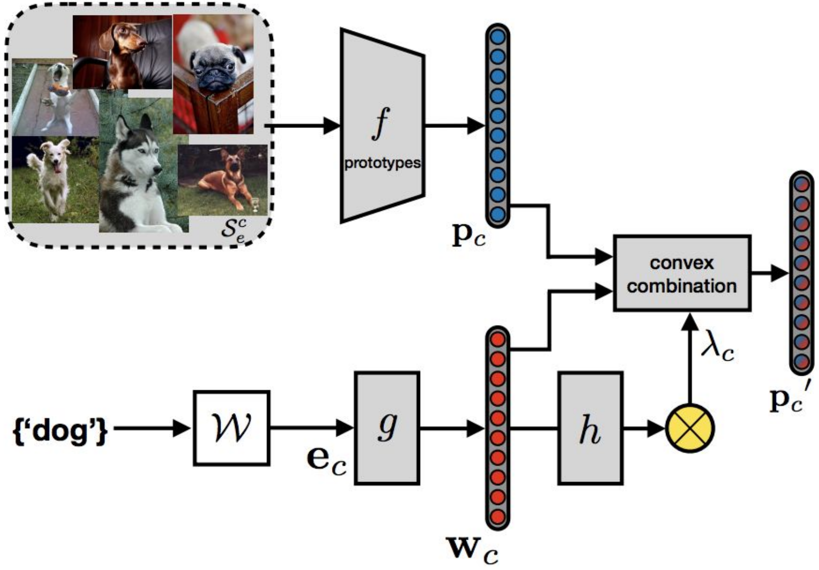
Doordash uses triplet network to train catalog item embeddings

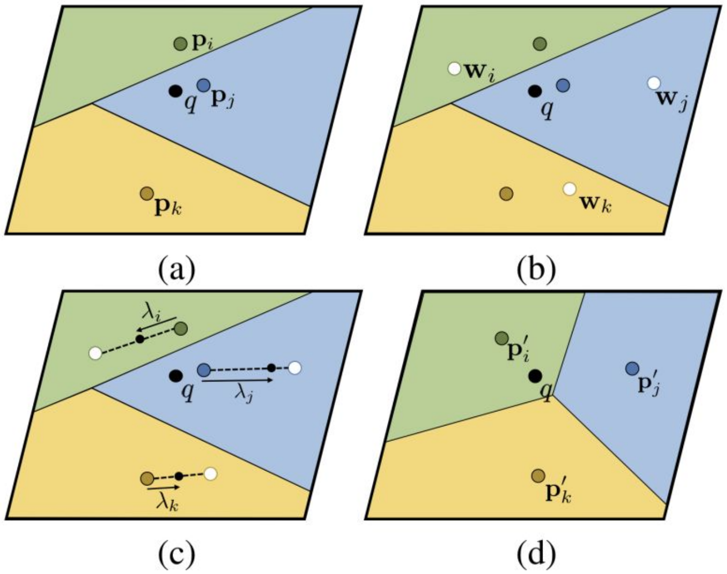




Source: Using Triplet Loss and Siamese Neural Networks to Train Catalog Item Embeddings

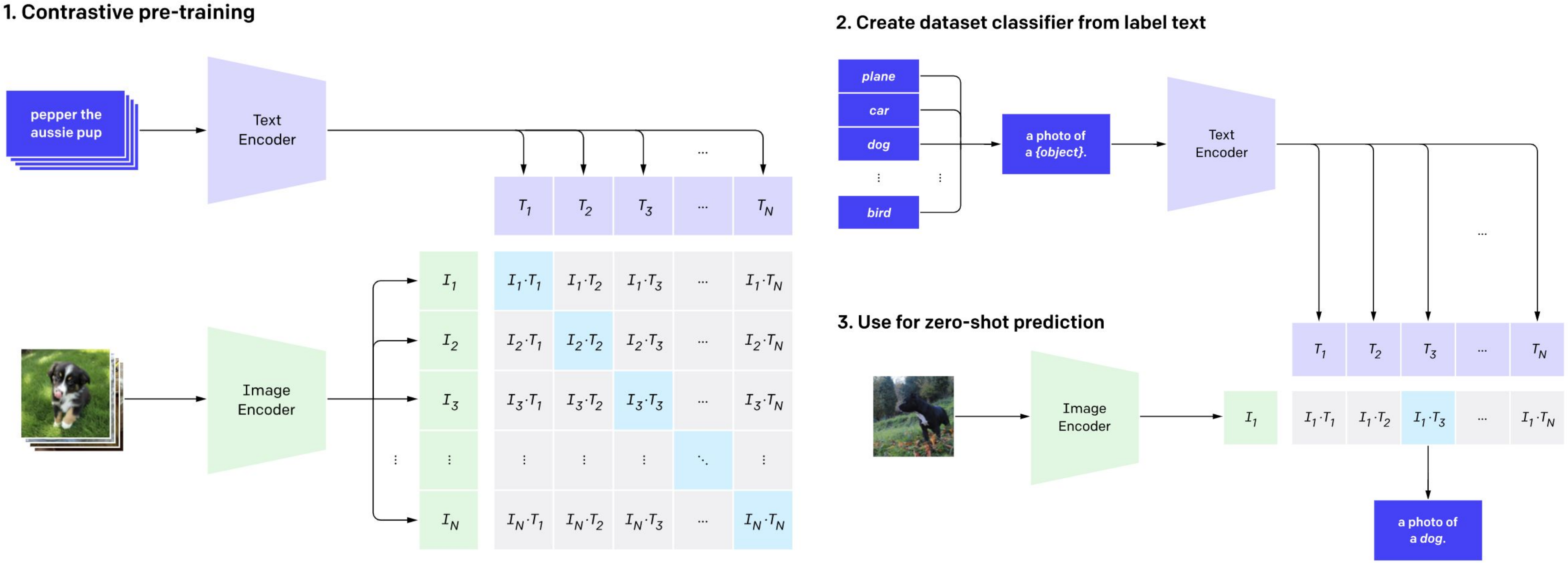
Adaptive Modality Mixture Mechanism (AM3) adaptively and selectively combines information from two modalities, visual and semantic, for few-shot learning.





Source: Adaptive Cross-Modal Few-shot Learning

OpenAI released Contrastive Language-Image Pre-Training (CLIP) model which is built on zero-shot transfer, natural language supervision and multi-modal learning



Source: [GitHub - openai/CLIP: Contrastive Language-Image Pretraining]

Q&A

Lesson 2: Siamese Neural Networks, Prototypical Networks, and Different Loss Functions

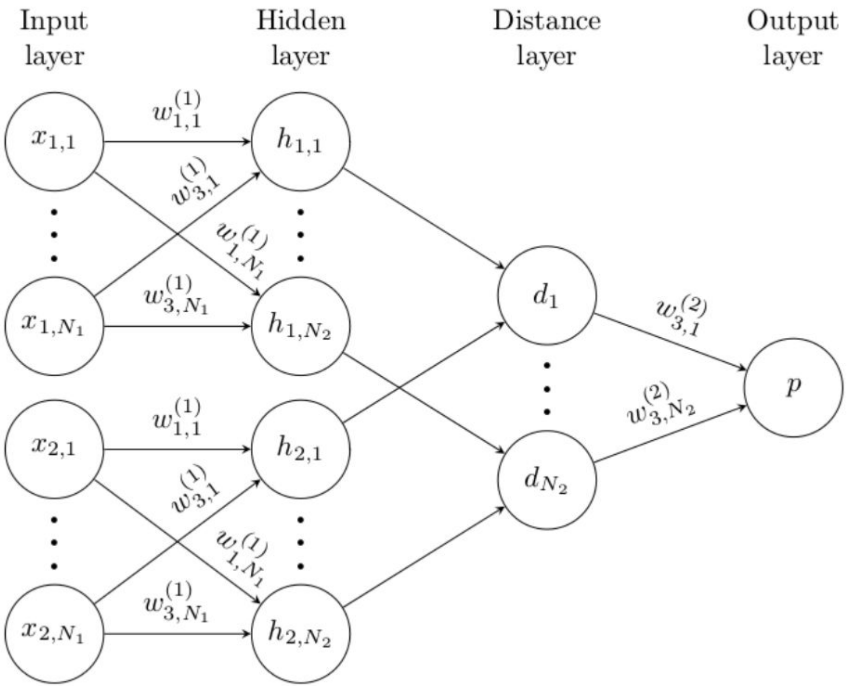
- Presentation:
  - Architecture of Siamese, Prototypical and Matching Networks (distance-based metric methods), concepts of Triplet Loss, Pairwise Loss, and transfer learning-based FSL.
  - Practical applications of these network architectures and some of the SOTA in FSL*
- Discussion: Open discussion on different loss functions and their pros/cons in terms of practical application.
- Presentation via Exercise:
  - Implementation of FSL in Natural Language Understanding (NLU) space using Siamese Network for text classification (Text classification via Siamese Network architecture)
  - Challenges involved in creating a dataset for the contrastive learning framework and the different approaches to creating the dataset.
- Q&A

Break (10 minutes)

* Revisit Lesson 1 Discussion Slides

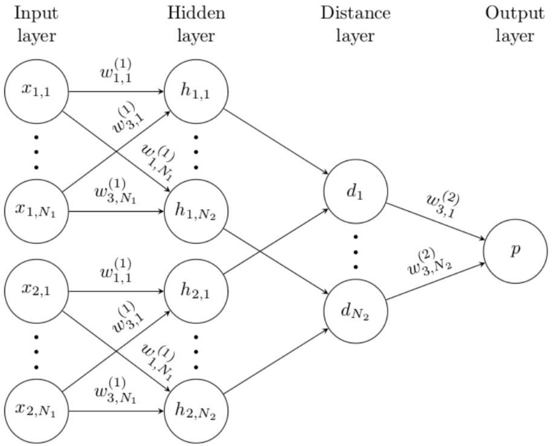
Siamese Neural Network is a non-parametric method

- A simple 2 hidden layer siamese network for binary classification with logistic prediction p.
- The structure of the network is replicated across the top and bottom sections to form twin networks (siamese network), with shared weight matrices at each layer.



Source: Siamese Neural Networks for One-shot Image Recognition

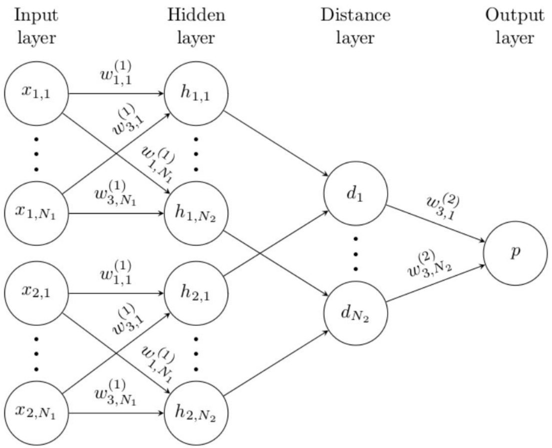
Siamese Network can be trained to learn if two images are of the same class or not, this can be a binary classification task



Label 0

Source: Siamese Neural Networks for One-shot Image Recognition

Siamese Network can be trained to learn if two images are of the same class or not



Label 1

Source: Siamese Neural Networks for One-shot Image Recognition

At test time, image in the query set is compared to the images in the support set

Watermelon Pineapple Grapes

Figure 1: The three card images comprising the support set

Figure 2: The card image comprising the query set

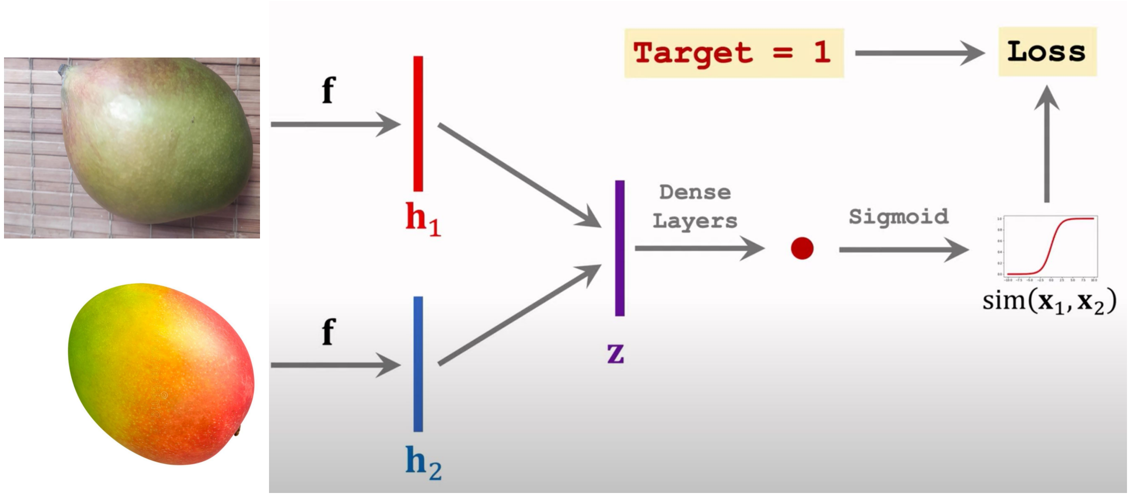
Meta-test time: compare image in Figure 2 to each image in Figure 1

Meta-training: Binary classification

Meta-testing: N-way classification

The Pairwise Loss function evaluates how well the network is distinguishing a given pair of images.

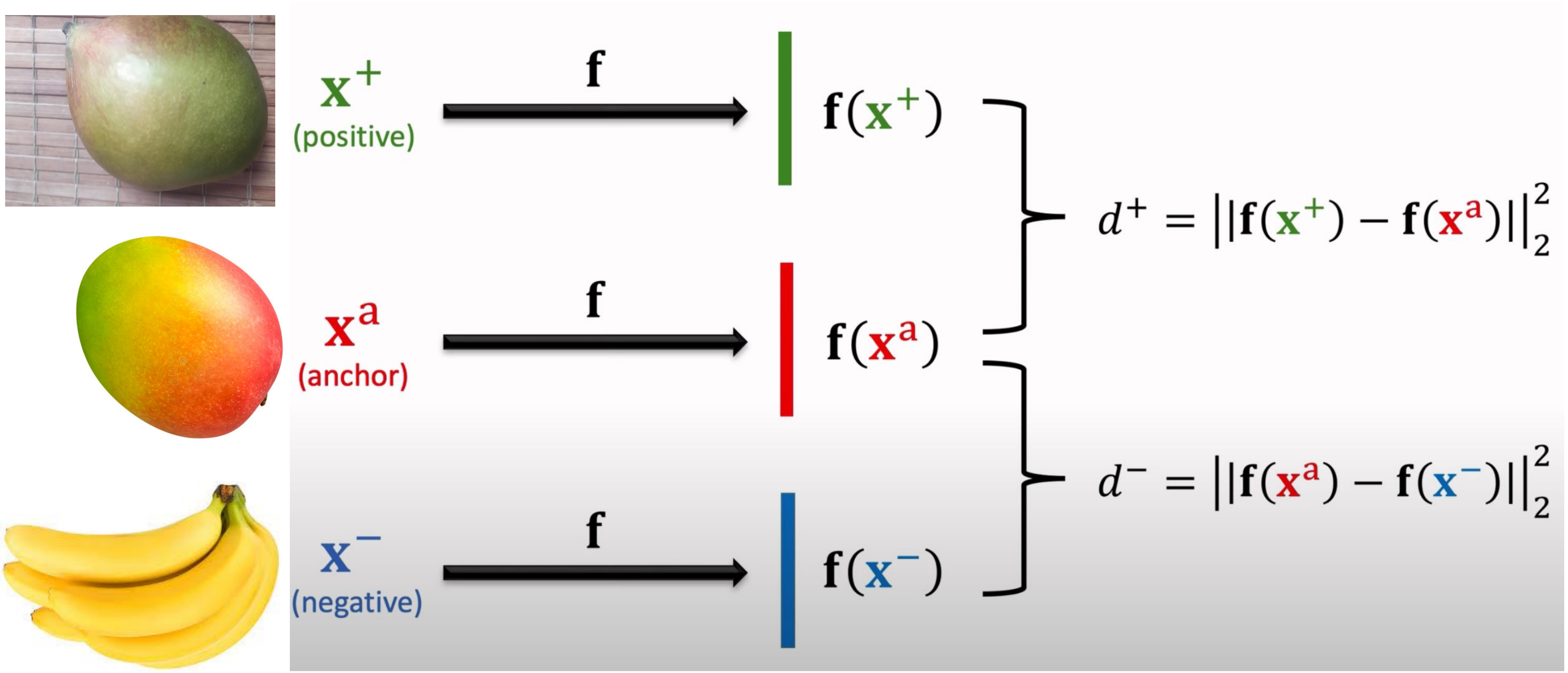
- Y: output label, it will be 1 if the image pairs are of the same class and 0 if the image pairs are from different classes.
- D: Euclidean distance between the output of the sister Siamese Network embeddings
- max(): takes the largest value between 0 and (margin - D).
- m: margin value which is greater than 0. Having a margin indicates that dissimilar pairs that are beyond this margin will not contribute to the loss. This makes sense, because you would only want to optimise the network based on pairs that are actually dissimilar, but the network thinks are fairly similar.



$$Y * D^2 + (1 - Y) * max(margin - D, 0)^2$$

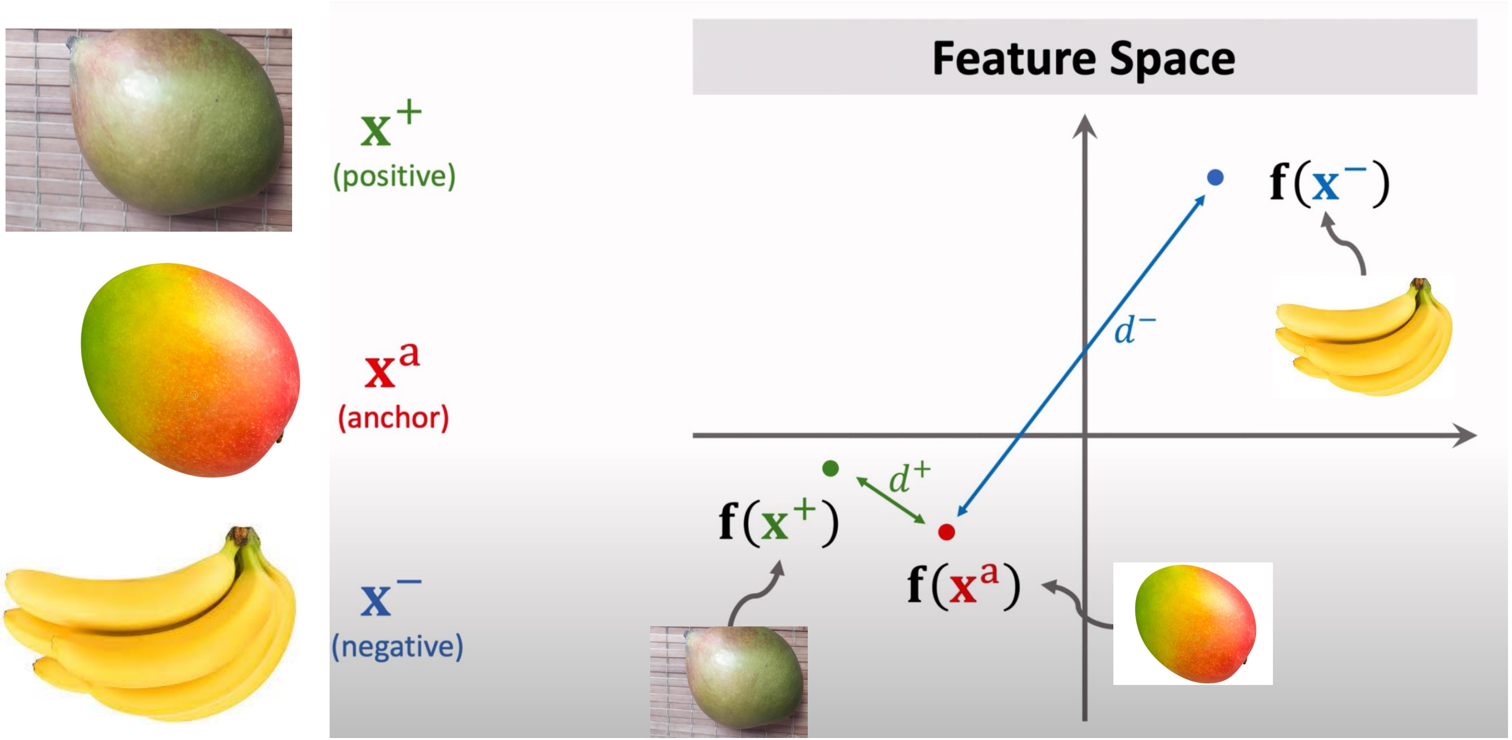
Source: https://pyimagesearch.com/2021/01/18/contrastive-loss-for-siamese-networks-with-keras-and-tensorflow

Triplet Loss is based on l2 norm distance between each class (positive & negative) and anchor



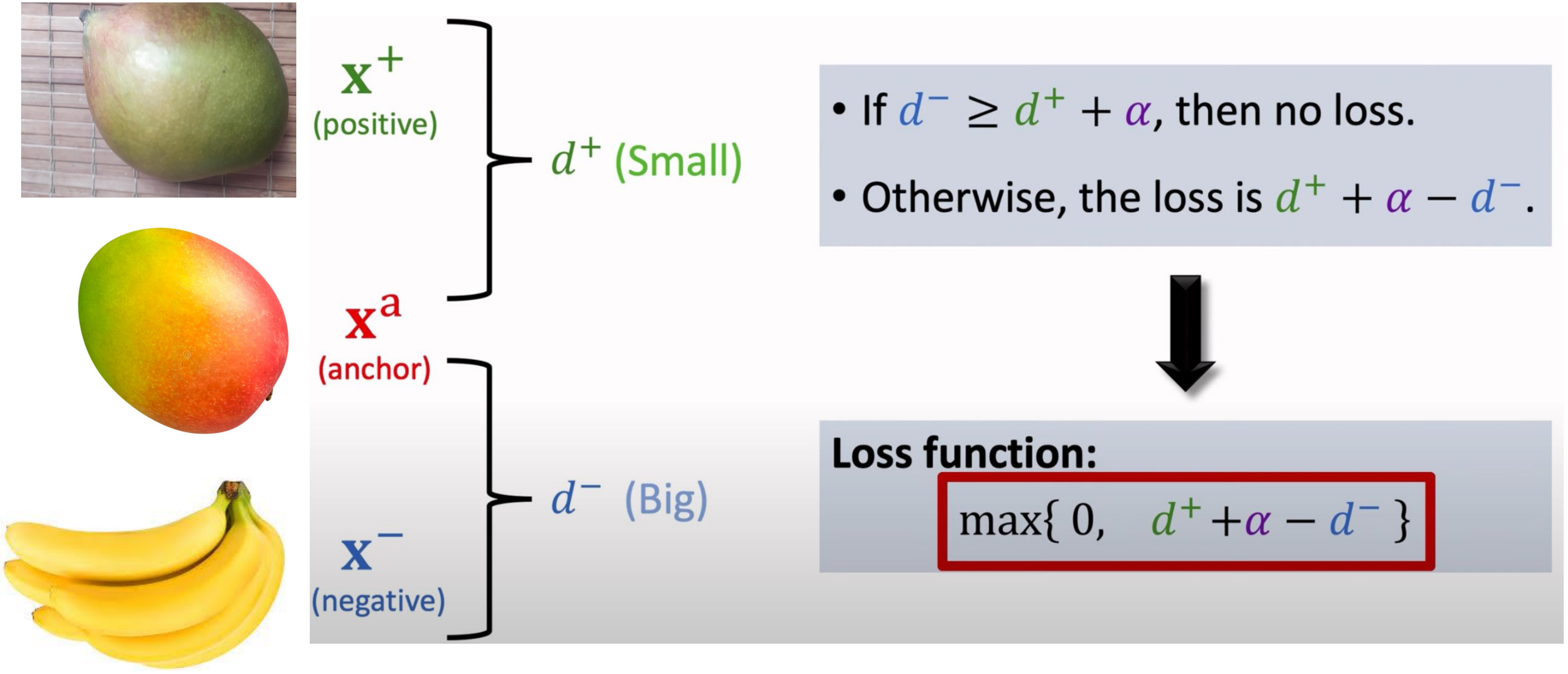
Source: Few-Shot Learning (2/3): Siamese Networks

Triplet Loss: the objective is that d+ is less than d-



Source: Few-Shot Learning (2/3): Siamese Networks

Triplet Loss is based on triplets of samples: positive, anchor, and negative



α (>0) is margin

Source: Few-Shot Learning (2/3): Siamese Networks

Prototypical networks is based on the idea that there exists an embedding in which points cluster around a single prototype representation for each class

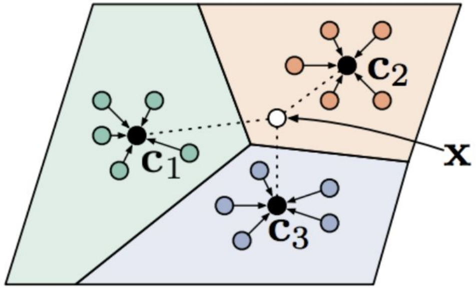
In few-shot classification we are given a small support set of N labeled examples $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ where each $\mathbf{x}_i \in \mathbb{R}^D$ is the D-dimensional feature vector of an example and $y_i \in \{1, \ldots, K\}$ is the corresponding label. S_k denotes the set of examples labeled with class k.

Prototypical networks compute an M-dimensional representation $\mathbf{c}_k \in \mathbb{R}^M$, or *prototype*, of each class through an embedding function $f_\phi : \mathbb{R}^D \to \mathbb{R}^M$ with learnable parameters ϕ. Each prototype is the mean vector of the embedded support points belonging to its class:

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i) \tag{1}$$

Given a distance function $d : \mathbb{R}^M \times \mathbb{R}^M \to [0, +\infty)$, prototypical networks produce a distribution over classes for a query point $\mathbf{x}$ based on a softmax over distances to the prototypes in the embedding space:

$$p_\phi(y = k \mid \mathbf{x}) = \frac{\exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_{k'}))} \tag{2}$$



d: Euclidean, or cosine distance

Source: Prototypical Networks for Few-shot Learning

Pseudocode to compute the loss J(φ) for a training episode for Prototypical Networks

- Learning proceeds by minimizing the negative log-probability $J(\varphi) = -\log p_\varphi(y = k \mid x)$ of the true class k via SGD.
- Training episodes are formed by randomly selecting a subset of classes from the training set, then choosing a subset of examples within each class to act as the support set and a subset of the remainder to serve as query points.
- Prototypical networks learn a metric space in which classification can be performed by computing distances to prototype representations of each class.

Algorithm 1 Training episode loss computation for prototypical networks. N is the number of examples in the training set, K is the number of classes in the training set, $N_C \leq K$ is the number of classes per episode, N_S is the number of support examples per class, N_Q is the number of query examples per class. $\textsc{RandomSample}(S, N)$ denotes a set of N elements chosen uniformly at random from set S, without replacement.

Input: Training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where each $y_i \in \{1, \ldots, K\}$. $\mathcal{D}_k$ denotes the subset of $\mathcal{D}$ containing all elements $(\mathbf{x}_i, y_i)$ such that $y_i = k$.

Output: The loss J for a randomly generated training episode.

$V \leftarrow \textsc{RandomSample}(\{1, \ldots, K\}, N_C)$ ▷ Select class indices for episode

for k in $\{1, \ldots, N_C\}$ **do**

  $S_k \leftarrow \textsc{RandomSample}(\mathcal{D}_{V_k}, N_S)$ ▷ Select support examples

  $Q_k \leftarrow \textsc{RandomSample}(\mathcal{D}_{V_k} \setminus S_k, N_Q)$ ▷ Select query examples

  $\mathbf{c}_k \leftarrow \frac{1}{N_C} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i)$ ▷ Compute prototype from support examples

end for

$J \leftarrow 0$ ▷ Initialize loss

for k in $\{1, \ldots, N_C\}$ **do**

  for $(\mathbf{x}, y)$ in Q_k **do**

    $J \leftarrow J + \frac{1}{N_C N_Q}\left[d(f_\phi(\mathbf{x}), \mathbf{c}_k)) + \log \sum_{k'} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k)) \right]$ ▷ Update loss

  end for

end for

Source: Prototypical Networks for Few-shot Learning

Prototypical Networks can be used in both few-shot and zero-shot scenarios

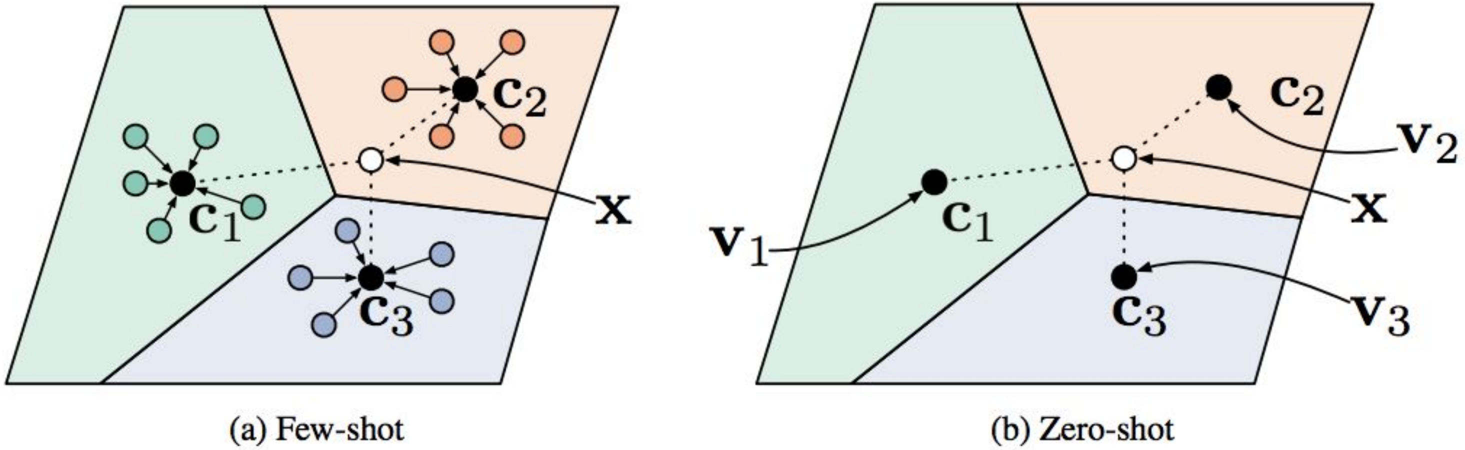


Figure 1: Prototypical networks in the few-shot and zero-shot scenarios. **Left**: Few-shot prototypes $\mathbf{c}_k$ are computed as the mean of embedded support examples for each class. **Right**: Zero-shot prototypes $\mathbf{c}_k$ are produced by embedding class meta-data $\mathbf{v}_k$. In either case, embedded query points are classified via a softmax over distances to class prototypes: $p_\phi(y = k|\mathbf{x}) \propto \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))$.

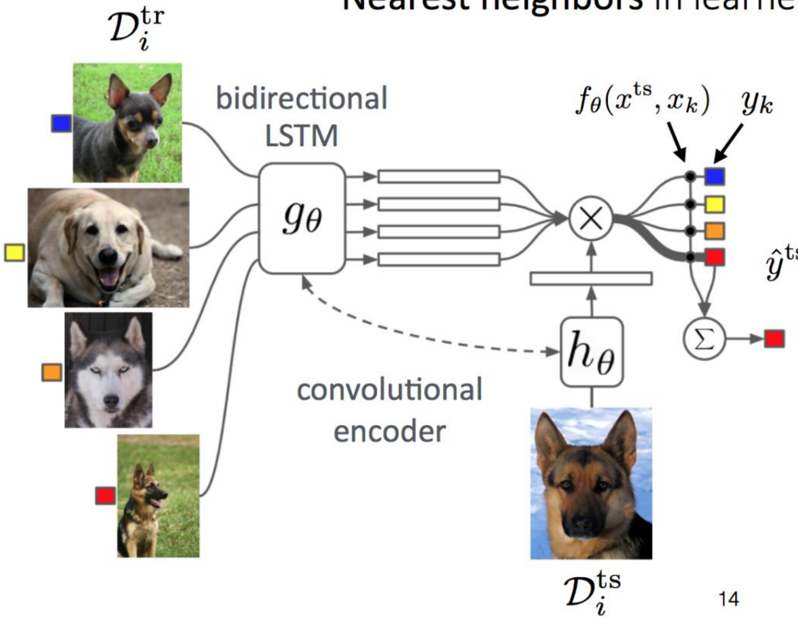
d: Euclidean, or cosine distance

Source: Prototypical Networks for Few-shot Learning

Matching networks produce a weighted nearest neighbour classifier given the support set

Can we **match** meta-train & meta-test?

Nearest neighbors in learned embedding space



$$\hat{y}^{\mathrm{ts}} = \sum_{x_k, y_k \in \mathcal{D}^{\mathrm{tr}}} f_\theta(x^{\mathrm{ts}}, x_k) y_k$$

Trained **end-to-end.**

Meta-train & **meta-test** time **match.**

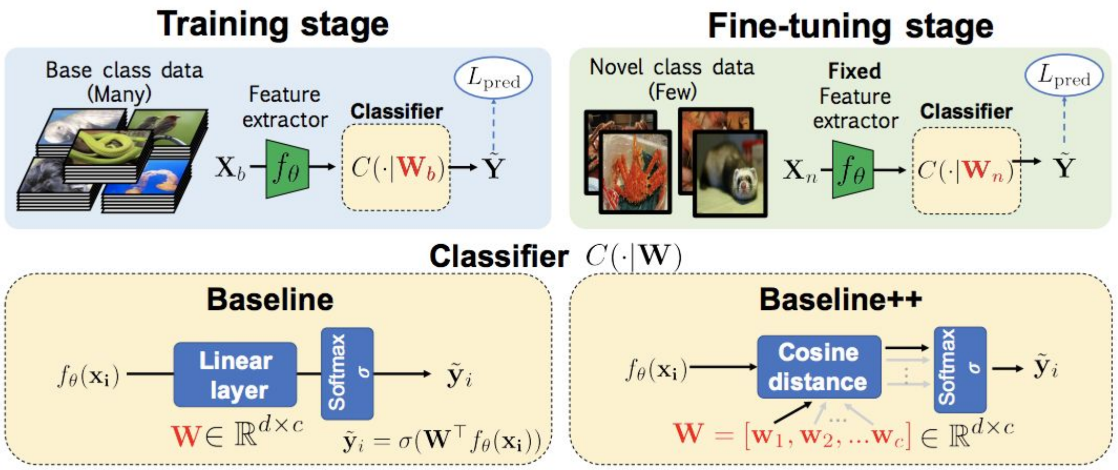
14

Vinyals et al. Matching Networks, NeurIPS '16

Source: https://cs330.stanford.edu/

Baseline model follows the standard transfer learning procedure of network pre-training and fine-tuning

- Both the baseline and baseline++ method train a feature extractor fθ and classifier C(.|Wb) with base class data in the training stage.
- In the fine-tuning stage, we fix the network parameters θ in the feature extractor fθ and train a new classifier C(.|Wn) with the given labeled examples in novel classes.
- The baseline++ method differs from the baseline model in the use of cosine distances between the input feature and the weight vector for each class that aims to reduce intra-class variations.



Source: A CLOSER LOOK AT FEW-SHOT CLASSIFICATION

Group Discussion

Different loss functions and their pros/cons in terms of practical application

Pairwise and Triplet Loss methods are robust to class imbalance but can require longer training time than the traditional supervised training

Pros	Cons
Fewer data samples required	Large amount of training data because of so many pairs of classes
Can works with high imbalanced data	Can require longer training than traditional supervised training methods (As the number of training samples increases, the number of image pairs and triplets grows drastically making it difficult to train on all possible pairs or triplets)
Helps in learning embeddings that place the same classes/concepts close together. Hence, we can learn semantic similarity.	Poor choice of training pairs and triplets i.e easy samples can lead to ineffective learning of discriminative feature (hard negative mining)

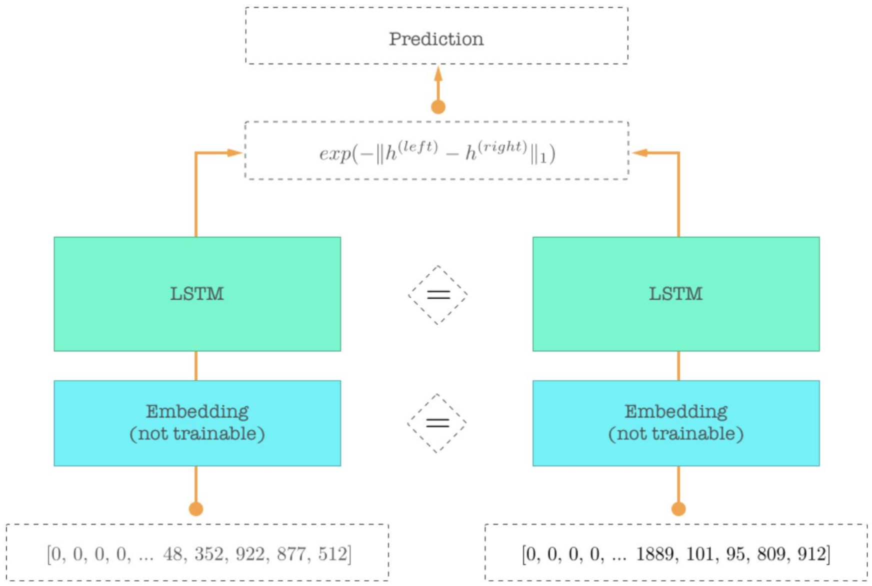
Source: A Friendly Introduction to Siamese Networks | Built In, Neural Networks Intuitions: 9. Distance Metric Learning | by Raghul Asokan | Towards Data Science

Exercise study of implementation of Siamese Network for text classification

- Notebook: Text classification via Siamese Network architecture (Can open locally or on Google Colab)

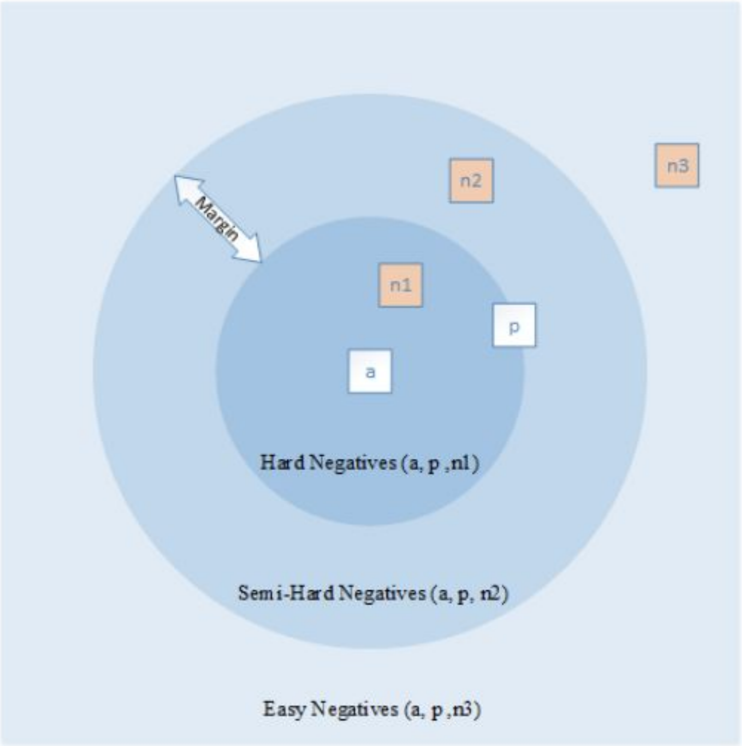
- MaLSTM's architecture (Ma for Manhattan Distance): Similar color means the weights are shared between the same-colored elements.

- In the above exercise, binary cross entropy is used instead of contrastive loss - any potential issues with that?



Source: How to predict Quora Question Pairs using Siamese Manhattan LSTM, Learning Sentence Similarity with Siamese Recurrent Architectures

The sampling strategy is capable of increasing both the success and the training time of the contrastive network



Hard Negative Mining

$$d(a,n) < d(a,p)$$

Semi-Hard Negative Mining

$$d(a,p) < d(a,n) < d(a,p) + margin$$

Easy Negative Mining

$$d(a,p) + margin < d(a,n)$$

Source: Deep Metric Learning: A Survey

Q&A

Lesson 3: Implementation of Siamese Neural Networks with Triplet Loss on Imagery Data (Code Work)

- Exercise:
  - Learn the implementation of Siamese Neural Networks by digging into the architecture code, and pick up hands-on skills to train these networks on the Imagery dataset, to learn similarities and differences between images. Learn how to prepare datasets for FSL training, and how to perform inference (dataset). (Siamese_Network_for_image_similarity)
- Poll (5 minutes): Comprehensive & Exercise check-in
- Recap of concepts learned today
- Q&A and Wrap Up

Poll

Comprehensive & Exercise Check-in

Which one of the following statements are true? (Multiple Choice Question)

- Language modeling effectively results in meta-learning because it seems to teach model about word meanings, syntax, grammar, world knowledge, and how to perform tasks.
- In zero-shot learning, the model predicts the answer given only a natural language description of the task.
- The triplet loss attempts to force similar examples together and push dissimilar examples apart in the latent space.
- Transfer learning does not always imply that the novel classes have very-few samples.
- In the case of one-shot learning, since there is only one support point per class, matching networks and prototypical networks become equivalent.
- To train a model using pairwise loss function, we would have to prepare dataset containing pairs of similar and dissimilar images.
- FSL can help to gather training samples from unlabeled data and learn a representation for the label.

Recap of concepts learned today

- Lesson 1: Fundamentals of Few Shot Learning
    - Basic concepts of FSL, meta-learning framework (support query framework).
    - Difference between supervised learning and FSL, difference between few-shot, one-shot and zero-shot learning.

- Lesson 2: Siamese Neural Networks, Prototypical Networks & different loss functions
    - Architecture walk-through.
    - Implementation of FSL in Natural Language Understanding (NLU) space using Siamese Network for text classification through an exercise.
    - Different ways to create dataset for contrastive learning framework

- Lesson 3: Implementation of Siamese Neural Networks with Triplet Loss using open source Imagery Data

Q&A

Thank You!

Instructor Contact: www.linkedin.com/in/ishachaturvedi93

O'REILLY®