

**Deep Learning for Computer Vision**  
**Professor. Vineeth N Balasubramanian**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Hyderabad**  
**Lecture No. 73**  
**Few Shot and Zero Shot Learning – Part 01**

In this last week of lectures, we will discuss a few advanced topics related to Deep Learning for Computer Vision. We will start with a contemporary topic, Zero shot and Few shot learning.

(Refer Slide Time: 00:32)


The slide content includes the NPTEL logo, a 'Review' header, a 'Question' box containing a bullet point: 'Last lecture, we saw methods that use videos as input for generating videos. Can we generate a video from a single image? See [this article](#) and [this paper](#); it uses single-shot learning, which is what this lecture is about'. The footer features a small photo of the speaker, the text 'Vineeth N B (IIT-H)', the slide title '§12.1 Zero-shot and Few-shot Learning', and the page number '2 / 33'.

Before we move forward, the question that we left behind in the last lecture on Applications for Deep Generative Models to Video understanding, we saw methods that use videos as input for generating videos. The question that we left behind was, can we generate a video from a Single Image? You may have guessed the answer. The answer is yes, we are not going to discuss the specific method now. But you can see the article linked on the slide and the corresponding paper relevant to the article.

The core idea of doing this is through a method known as Single Shot learning, where using a single instance, which in our case is an image, the network generates or learns to generate a complete video. Interestingly, single shot learning is one specific setting of few shot learning, and is the focus of this lecture. So, this becomes a nice segue into what we are going to discuss in

this lecture. But for more details, please come back and visit this article to understand how a video can be generated from a single image.

(Refer Slide Time: 02:05)



### Learning with Limited Supervision

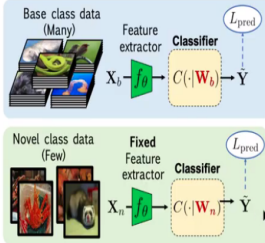
**Problem**

- Deep learning models  $\implies$  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



The diagram illustrates two scenarios. The top scenario, 'Base class data (Many)', shows a 'Feature extractor' receiving input  $X_b$  and outputting features  $f_b$ , which are then processed by a 'Classifier'  $C(\cdot; W_b)$  to produce output  $\hat{Y}$ . The bottom scenario, 'Novel class data (Few)', shows a 'Fixed Feature extractor' receiving input  $X_n$  and outputting features  $f_n$ , which are processed by a 'Classifier'  $C(\cdot; W_n)$  to produce output  $\hat{Y}$ . Both scenarios include a loss function  $L_{pred}$ .



Vineeth N B (IIT-H)
§12.1 Zero-shot and Few-shot Learning
3 / 33



Deep learning models are known to be heavily reliant on large amounts of labeled data during training. That is one of the reasons they work so well. But that is also one of their major limitations. If you had only a few samples to learn from, deep learning models may not be that effective in giving you a good model, and a good prediction performance. To a certain extent, regularization methods are used to avoid over fitting in low data regimes, but they do not really give you good performance on low data regimes. So, what do we do?

The solution is to be able to train models explicitly, that are capable to rapidly generalize to new tasks with only a few samples, or perhaps no samples at all. And that is the focus of this lecture. The objective is to enable models to perform under practical settings, where data annotation could be infeasible, could be expensive, or new classes get added over time. So, illustrate illustratively speaking, you have a few what are known as base classes, where you may have many examples.

The goal is to use them and learn a classifier in such a way that if you have other classes, for which you may have very few samples, or no samples at all, then the classifier can still work and

be able to give you a good prediction. Let us see how this can be done over the rest of this lecture.

(Refer Slide Time: 04:04)

### 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,  $Y^u \cap Y^s = \emptyset$



Vineeth N B (IIT-H)
§12.1 Zero-shot and Few-shot Learning
4 / 33

So, this problem setting is called Few Shot learning or Zero shot learning. Let us review the formulation first for both of these, if  $x$  was an image or a feature, you could assume that the input to these models need not be an image per se, it could just be say a VGG feature or a ResNet feature, which is obtained by passing the image through a VGG network or a ResNet network for that matter. And  $y$  denotes the class label for that data point, then, Few shot learning is formalized as, you have training data,  $D_{train}$  given by your  $(x_i, y_i)$  tuples. Let us assume there are total of  $I$  tuples, where for some of the classes there are only a few training samples.

Few shot learning is also characterized as an N-way-K-shot setting, for where for  $N$  of your total number of classes, if you had total number of classes to be, say capital  $C$ , for  $N$  of those,  $N$  less than  $C$ , you have only  $K$  samples, while for the rest of the classes, you have a lot of samples. The rest of the classes are also called base classes, as you saw on the slide, in the earlier, on the earlier slide. That is a Few shot learning.

What is Zero shot learning? Zero shot learning is when few becomes 0, what does that mean? You have training data,  $x, y$ . But now, because you want to be able to classify a data point that was never in your training set, that is the zero shot setting where you do not have for some of

your classes, you have reasonable amounts of data for some of the classes, which are also called zero shot classes, there is absolutely no label data at all.

In such a setting, your training data consists of  $x$ ,  $y$ , and  $a(y)$ , which are some attributes related to each class  $y$ . It could be some textual description for instance, so if you had images of all animals, Let us say you have seen a Lion, Tiger, a Zebra, but you have never seen a horse. Let us say that image was never available to you. But you know how the description of a horse looks like in text or as a set of attributes. When we say a set of attributes, we mean whether the animal has a tail, 4 legs, what colors can it assume, does it have a mane, so on and so forth. So, those are what we refer to as attributes  $a(y)$ , you can look at them as metadata that are provided, even if the data is not available.

So, that is the zero shot setting where  $x \in X^s$ , which is known as seen classes,  $y \in Y^s$ , which are again the Labels of the seen classes, and  $a(y)$  are a set of attributes for all classes. And the test time here in zero shot is a setting where you have  $x$ ,  $y$ ,  $a(y)$ , where  $x$  comes from an unseen class  $X^u$ ,  $y$  is the corresponding label and  $a(y)$  is the corresponding attribute. Importantly,  $Y^s \cap Y^u$  is the null set. So, the seen classes and unseen classes have no intersection.

(Refer Slide Time: 08:12)



### 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  $f : x \rightarrow Y^s \cup Y^u$
- 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





Within few shot/zero shot, there are two further ways of categorizing them. One of them is what is known as Conventional few shot or zero shot, where in the problem setting, the goal is to learn

a classifier  $f$  such that the image or feature  $x$  to be recognized at test time comes only from the unseen or Few shot classes, not from the base classes, or the scene classes. That is what is known as a conventional few shot or zero shot. That is how the initial methods in the space were developed about 4 or 5 years ago.

The other setting, which is the more challenging and more practical setting is Generalized Zero shot and few-shot learning, where the goal is to still learn a classifier going from  $x$  to both seen and unseen classes or base and few-shot classes. Where the image or feature  $x$  to be recognized at test time may belong to either seen or unseen, or base or Few shot.

Clearly, the generalized zero-shot or few-shot learning setting is the more challenging and more useful one. Because in the real world, you cannot predict whether your image at test time will come only from an unseen class or only from a few shot class. It could come from any class in your universe of classes.

(Refer Slide Time: 09:59)


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

Credit: Wang et al, Generalizing from a Few Examples: A Survey on Few-Shot Learning, ACM Computing Surveys'20  
 Vineeth N B (IIT-H) §12.1 Zero-shot and Few-shot Learning 6 / 33




At this juncture, Let us recall one of the fundamental tenets of Supervised Machine Learning, which is Empirical Risk Minimization. If  $p(x, y)$  is a ground truth joint probability distribution of input  $x$  and output  $y$ . The goal of supervised learning is given a hypothesis  $h$  or a model  $h$ , the goal is to minimize the expected risk or loss measured with respect to the joint probability distribution, which is then given by  $R(h)$  is integration of the loss, the loss is has two inputs,  $h(x)$

, the prediction of the model, and  $y$ , which is the expected output. So, the loss is computed between these two quantities.

And you integrate this over the entire joint probability distribution, which is also denoted as the expectation over the distribution of the loss. Unfortunately, in the real world, we cannot estimate the joint probability distribution. So, this expectation is approximated as the empirical risk with respect to the training data that is provided to us. So, the empirical risk is given by

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

(Refer Slide Time: 11:41)



### 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  $H$

Credit: Wang et al, Generalizing from a Few Examples: A Survey on Few-Shot Learning, ACM Computing Surveys'20



Vineeth N B (IIT-H)
§12.1 Zero-shot and Few-shot Learning
7 / 33

Now, Let us assume that we have a family of models, you could assume we are solving a linear problem. So, you could assume that, if it was a linear SVM, your family of models belong to lines. So, that is what we denote by  $H$  depends on what algorithm what choice you are making. But let us now assume that the family of models that you are trying to search for to address this problem is given by capital  $H$ .



And let one of the hypothesis of the models be small  $h$ . Then we have 3 kinds of functions. One is  $\hat{h}$ , which is the original function that minimizes expected risk that we are really looking for, that is the ground truth function. Then we have  $h^*$ , which is the function belonging to the family that we are searching for, within this family capital  $H$ , that minimizes the expected risk. And

finally, we have  $h_I$ , which is the function in  $H$  that maximizes the empirical risk, which is the risk of the loss with respect to your training data.

Given these 3 terms, one could now define the gap or the generalization gap between what our model with respect to the training data has learned, which is  $h_I$  with respect to the original ground truth model,  $\hat{h}$ . The loss between them can be decomposed into two parts one  $R(h^*) - R(\hat{h})$ . If you chose this family of models in capital  $H$ , and the best model in that to be  $H^*$ , how close would that be to the actual model that could belong to any family of models. It need not be linear model, need not be a quadratic model, it could be any model. So,  $\hat{h}$  is any model,  $h^*$  is the best model. In this family of models that we have chosen hypothesis and model here mean the same thing.

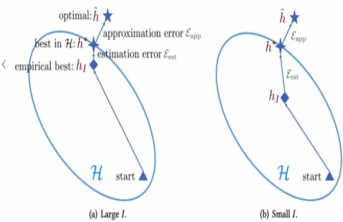
And the second term is how close  $h_I$  which is the model that we obtain from our training data is close to  $h^*$ , which is the ideal function which in the family of functions. So, the first component is denoted as approximation error, which measures how closely functions in this family  $H$  can approximate your optimal hypothesis  $\hat{h}$ . And the second term here is the estimation error, which measures the effect of minimizing the empirical risk with respect to the expected risk within the family of functions  $H$ . Why are we talking about all of this now?

(Refer Slide Time: 14:55)

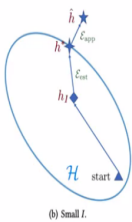



### Problems with Few/Zero-shot Setting

- $\mathcal{E}_{est} \implies$  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  $I$ .

#### 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

Comparison of learning with sufficient and few training samples

Credit: Wang et al, Generalizing from a Few Examples: A Survey on Few-Shot Learning, ACM Computing Surveys'20



Vineeth N B (IIT-H)
§12.1 Zero-shot and Few-shot Learning
8 / 33

We will come to that in the context of few shot and zero shot learning. The second term in the previous slide can be reduced. If you had large amounts of data, then your empirical risk minimizer becomes close to your expected risk minimizer. Because the training data when you have large amounts, resembles, the overall joint probability distribution more closely. See, if you had sufficient labeled data to train with, then your empirical risk minimizer gives a good approximation to the best possible  $h^*$ .

Unfortunately, with few-shot learning, and zero-shot learning, we do not have the sufficient amount of training data, especially for a certain set of classes. So, what is the problem? Let us look at this illustration. So on the left, what you have now is when you have a large number of data samples. You have  $\hat{h}$ , which is your optimal hypothesis denoted by the star here, then you have this ellipse, which represents the family of models capital H, that you are focusing on to ask your machine learning algorithm to learn or pick one of them.

So, the best solution there would be the projection of  $\hat{h}$  onto this family of models, which we denote as  $h^*$ . And finally, you now start your machine learning algorithm, somewhere here where the triangle is shown. And over the learning procedure, you converge to a model, which is given by  $h_T$ , which is fairly close to  $h^*$ . So, if you have large number of samples, you get an  $h_T$  that is close to  $h^*$ .

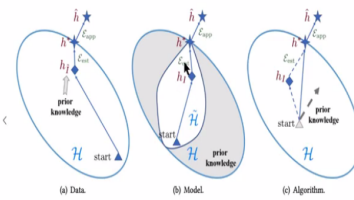
On the other hand, when you do not have a large number of samples, you may not be able to get very close to  $h^*$ , which is at least the best model that you can get in this family. So which means, the resultant  $R_T(h)$ , which is the empirical risk that you have from this particular model, is going to be far from being good a good approximation of the expected risk and the resulting model  $h_T$  may over fit to your training data.

(Refer Slide Time: 17:33)





## Addressing Few/Zero-shot Learning



Different perspectives of addressing few-shot learning

### Data

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

### Model

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

Credit: Wang et al, Generalizing from a Few Examples: A Survey on Few-Shot Learning, ACM Computing Surveys'20



Vineeth N B (IIT-H)

§12.1 Zero-shot and Few-shot Learning

9 / 33

How do you address this problem? So we can look at addressing this problem from 3 different perspectives: from a data perspective, from a model perspective and from an algorithm perspective. Let us now see each of them. From a data perspective, one could simply augment the training dataset, simply increase the number of samples to an  $\tilde{I}$ , which is far, far greater than  $I$ . And that should help you get a better estimate of the empirical risk minimizer. So, you can see here, that by increasing the number of samples, which we call prior knowledge here, we now push the  $h_I$  model closer to  $h^*$ , which is our ideal goal. And that would be by leveraging data.

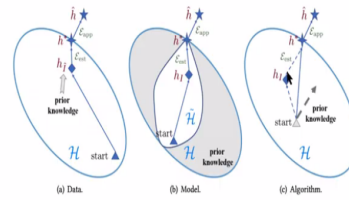
The second option is to do this by constraining the space of models to a smaller space, which is less likely to push  $h_I$  far away from  $h^*$ . So, you can see here that using some kind of a prior knowledge or domain understanding, we try to ensure that the space of hypothesis or models capital  $H$  is now restricted to only this non-shaded region in the center. And any model here is perhaps better than most of the models that are outside this non shaded space.

This is another way of addressing the setting where you may have only few samples from certain classes. What are we doing here? We are constraining the complexity of  $H$  to a much smaller hypothesis space, then you are hoping that even a small amount of training data will be sufficient to train a reliable  $h_I$  which is once again close to  $h^*$ .

(Refer Slide Time: 19:37)



## Addressing Few/Zero-shot Learning



Different perspectives of addressing few-shot learning

### Algorithm

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

Credit: Wang et al, Generalizing from a Few Examples: A Survey on Few-Shot Learning, ACM Computing Surveys '20



Vineeth N B (IIT-H)

§12.1 Zero-shot and Few-shot Learning

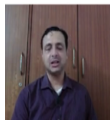
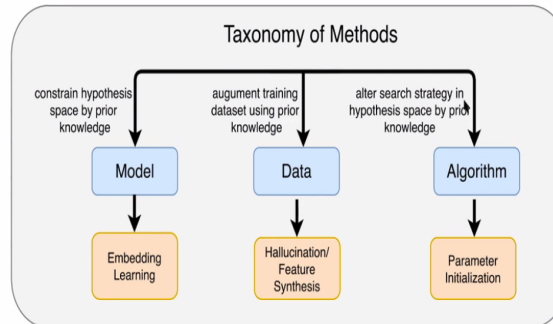
10 / 33

The third approach is to address this from an algorithmic perspective. We aim to search for parameters  $\theta$  of the best hypothesis  $h^*$  in capital  $H$ . One way this can be achieved is by using some kind of prior knowledge. We will see examples of methods later in this lecture, which alters the search strategy, and can also help provide a good initialization. So, you can see here that using some kind of a prior knowledge, you now have a start point, which is much better than the start point that you had in this figure a, which helps you converge to a better  $h_j$ . Each of these 3 approaches are valid approaches to address this problem. Let us see methods belonging to each of these kinds in this lecture.

(Refer Slide Time: 20:34)



## Taxonomy of Methods



Vineeth N B (IIT-H)

§12.1 Zero-shot and Few-shot Learning

11 / 33

So, from an overall taxonomy perspective, one could look at constraining the hypothesis space by prior knowledge, which is the model approach to solving few-shot, zero-shot learning. Examples of methods here are known as embedding learning methods. The second category of methods that are based on augmenting training Data the data approach are broadly grouped as Hallucination or feature synthesis methods. And the third case of altering the search strategy in hypothesis space by using some prior knowledge, the algorithmic approach are also sometimes categorized as Parameter Initialization methods. Let us see each of them in detail now.