FSL : Few Shot learning

- Armadilo :
- Pangolin :

} Almost similar to look at but yet are different.

- Support set : → at most a few samples. too small to train DNN
- Query : → can only provide few information at the test time.

Better learning Jargon :

The goal here is to learn to learn.

Goal to recognise the similarity to or difference.

model can know similarity and difference.

Meta learning → learning to learn by itself.

- Similarity & difference

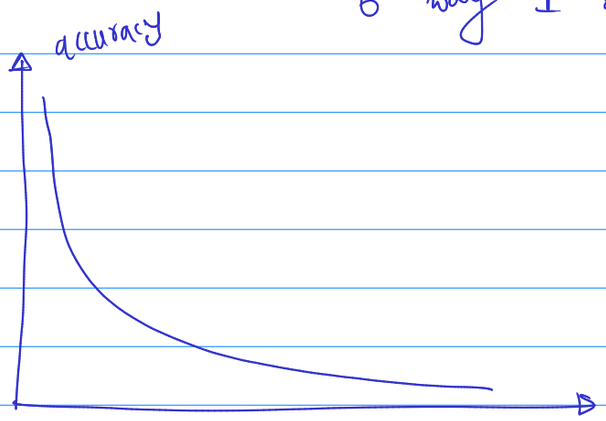
- One shot learning.

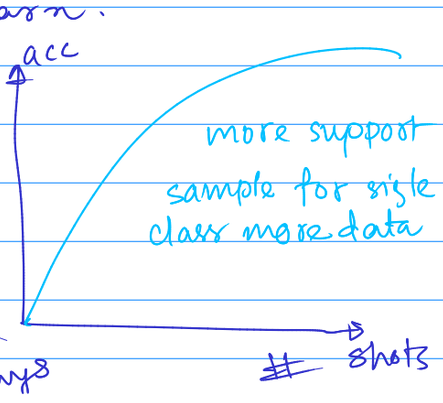
- Test Sample: from known class
  - Query :
  - Support :

Squirel : Support set :
six classes:
otter !
Bever : !

- Support Set : K classes data
  n nos of samples for each class.

6 way 1 shot learn.



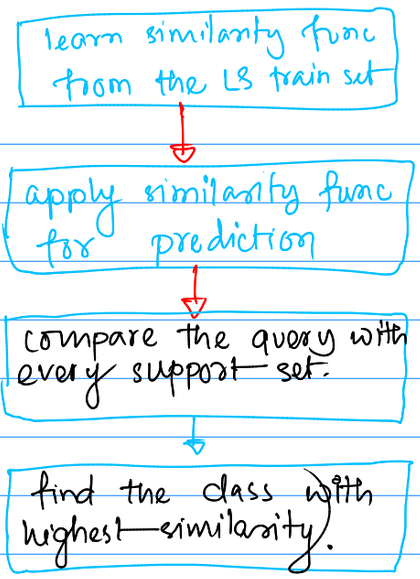


learn the similarity function $sim(x, x')$.

$sim(x_1, x_2) \approx 1$ if same

$sim(x_1, x_2) \approx 0$ if different.

Workflow:



omniglot: over 1000 classes — 1623 characters, 105 × 105 images!

But here only 10 samples per class.

Here alphabets from 50 languages are here.

50 alphabet

1623 unique characters

each characters written by 20 people

Samples are (105 × 105) pixels.

$= 1623 \times 20$

Total 32460 characters

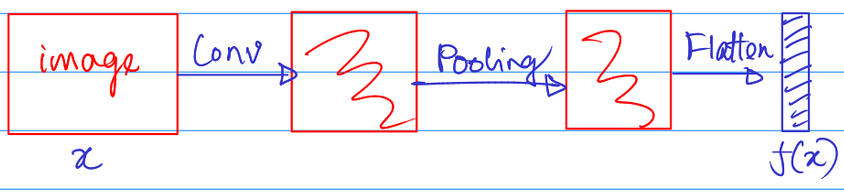
Subsets 19280 char (30 alph, 964 char)

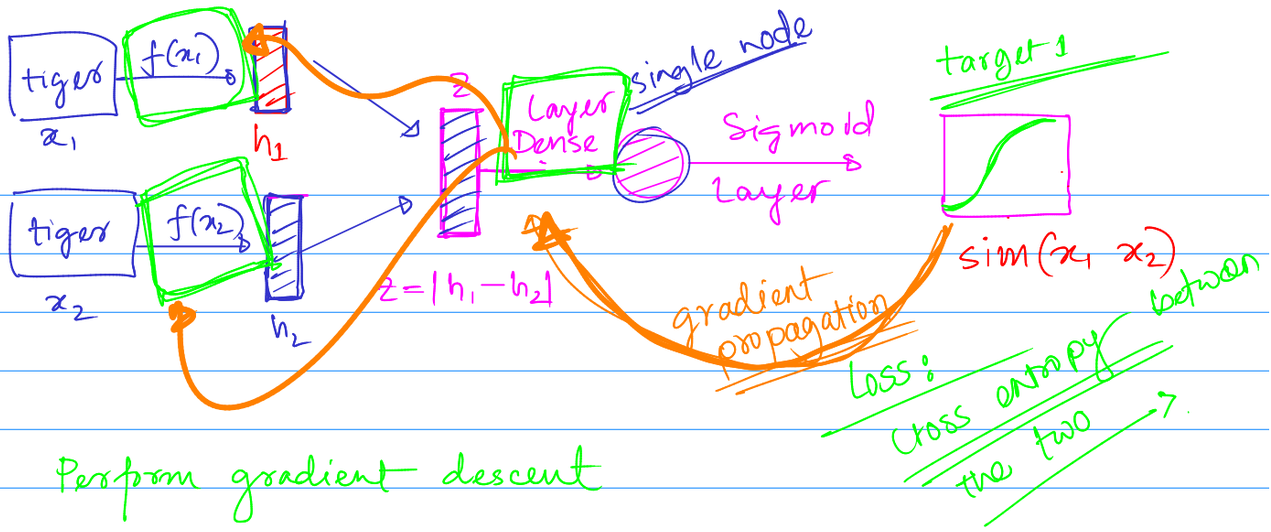
learning the useful feature extractors.

imageNet mini: 100 classes

Siamese Network:

Positive Samples	Negative Samples
$(x_1, x_2, 1)$ x_1 and x_2 comes from the same class	$(x_1, x_2, 0)$ x_1 and x_2 comes from the difference class.





Perform gradient descent

One shot Prediction:

6 class 1 shot are not in the training set.

S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}

q: Now get the similarity matrix between query and the support set using.

$sim(S_i, q)$ ✓

Triplet Loss.

Prepare training data:

each time select a random image from the images.

anchor + positive (x^+)

(x^a) − negative sample (x^-).

medium pen:

positive (x^+) $\xrightarrow{f}$ $f(x^+)$

anchor (x^a) $\xrightarrow{f}$ $f(x^a)$

negative (x^-) $\xrightarrow{f}$ $f(x^-)$

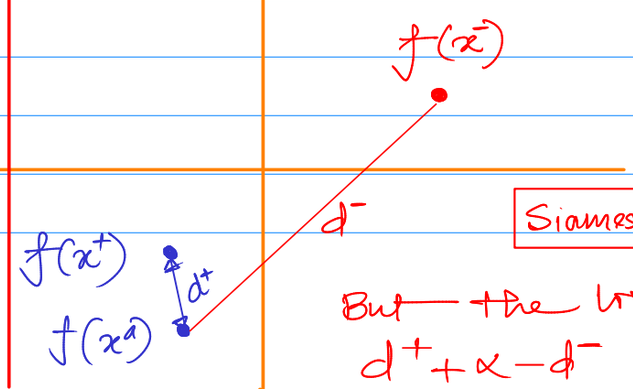
$d^+ = \|f(x^+) - f(x^a)\|_2^2$ should be small ✓ small

$d^- = \|f(x^-) - f(x^a)\|_2^2$ should be large. ✓ (large)

we use margin $\alpha > 0$

We hope $d^- \geq d^+ + \alpha$

if this is true, then loss 0.



Siamese Network.

But the loss $d^+ + \alpha - d^-$

So the final loss function becomes:

$$d^- \geq d^+ + \alpha$$

$$\Rightarrow \boxed{d^+ + \alpha - d^- \leq 0}$$

if true then loss 0.

or the loss is $(d^+ + \alpha - d^+)$ given here

$$\max\{0, \boxed{d^+ - d^- + \alpha}\}$$

can not distinguish between them

this is called triplet loss.

For one shot prediction: query image: q

Support set s_1 s_2 - - - - s_K

distance $dist(s_1, q) =$

$dist(s_2, q) =$

feature extractor:

$$(dist(f(x), f(s_i)))$$

use the convolutional neural network $f(x)$ used earlier in the neural network as the feature extractor. So for images $f(x)$ provides the feature.

dist smallest — the model.

K way n shot

$\begin{bmatrix}1\\2\\3\end{bmatrix}$ $\begin{bmatrix}a\\b\\c\end{bmatrix}$

assume x and w are two unit vectors

$\|x\|_2 = 1$

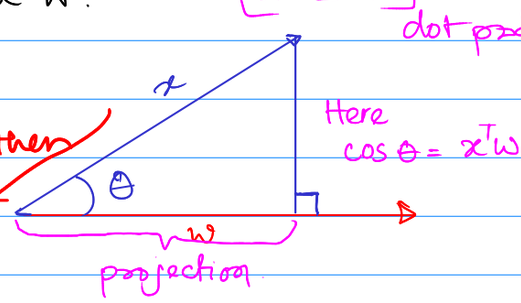
$\|w\|_2 = 1$

cosine similarity:

$\cos\theta = x^T w$.

$[1\ 2\ 3]$ $\begin{bmatrix}a\\b\\c\end{bmatrix}$

$[a+2b+c]$ dot product



Here if x and w are not unit vectors, then we can normalize them to be the unit vectors.

Then the cosine similarity is modified as follows

$$\cos\theta = \left(\frac{x}{\|x\|_2}\right)^T \frac{w}{\|w\|_2}$$

inner product ≡ cosine similarity

$$p = \text{softmax}(\phi) :$$

Entropy regularization:

- x is the sample
- $p = \text{Softmax}(W \cdot f(x) + b)$ is the prediction.
- Entropy $H(p) = -\sum_i p_i \log p_i$.
- for all the queries, average of $H(p)$ for all queries.
- Encourage the entropy to be small.

Bad
Big Entropy

good classifiers
confident classifier.
Good.
Low entropy.

Regular Softmax

$$p = \text{Softmax}(Wq + b)$$

$$= \text{Softmax}\left(\begin{bmatrix} w_1^T q + b_1 \\ w_2^T q + b_2 \\ w_3^T q + b_3 \end{bmatrix}\right)$$

Replacing the dot product with similarity we get

$$p = \text{Softmax}\begin{pmatrix} \text{sim}(w_1, q) + b_1 \\ \text{sim}(w_2, q) + b_2 \\ \text{sim}(w_3, q) + b_3 \end{pmatrix} \quad \text{where} \quad \text{sim}(w, q) = \frac{w^T q}{\|w\|_2 \|q\|_2}$$

- Fine tuning ✓
- Entropy Regularization. ✓
- Cosine + Softmax.

} great!