30/06/20 ॐ

Introspection

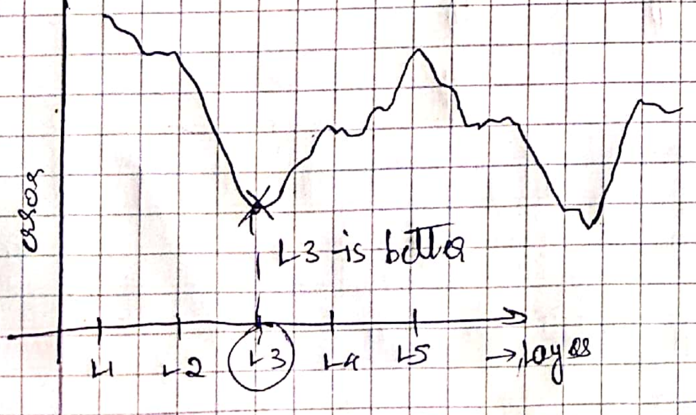
I) Basic methods

i) Compare classification metric at start / middle and end of training per o/p category

ii) Manually chek error / misclassification and figure out the reason
Ex:- blurring, noise, insta filters.

iii) Plot weight distributions and compare weight distros
→ Plot weight distribution of each layer and see how the differ, if they look the same then there might not be much of learning going on there.
→ if two distros look diffrent then some learning has happened
→ Outer layers needs smaller changes than inner layers.

iv) Linear classifier probe :- Train a linear classifier like logistic regression on intermediate layers. if some intermediate layer is more linearly separable than others then its ~~ca~~ activations can be used
→ This is especially useful for unsupervised pretraining as we don't know which layers representation is better



→ But cannot learn any other stuff that is outside of current data distribution

II) CNN Introspection

Types of introspection

a) Feature visulization :- Produce pattan characteristic for particular neuron.

b) Saliency map :- Trace back output To input (which input pixels are imp
(Attribution)

a) Feature visualization :- Find input which makes o/p more likely Ex:-Finding a Typical image for cat, Average face of a Man

→ Can look at output and optimize i/p To maximally activate it

→ Instead of optimizing weights To minimize The loss, we optimize inputs To minimize loss. This can be done at Layer/channel or neuron level.

New Input (neuron/channel/Layer) = Old Input (Layer/channel/Neuron) − Loss

** → This is done in googles deep dream and can be useful To know what exact patterns are our NN producing. But Deep dream lots of Time produces rubbish/non-intuitive patterns, hence This Technique is not much preferred

** → Also while optimizing input we Tend To get an unnatural/high frequency/brightly colored/checkerboard patterned optimal input. Then we need To use regularization strategies

i) Frequency penalization :- Motivate pixels To stay in some frequency range and heavily penalize high frequency

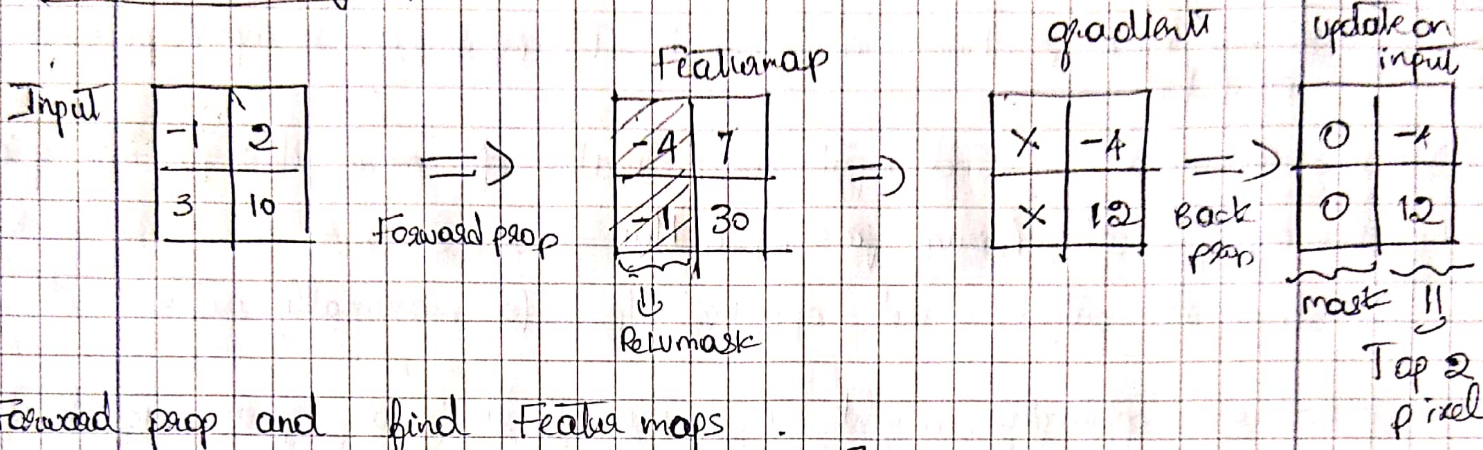
ii) Transformation robustness :- Makes sure That even affine Transformed optimal input yields similar result as most optimal input. Without This, even a single pixel change To most optimal input can completely flip The output classification (Similar To one-pixel attack adversarial)

iii) Learned prior :- Forcing model To produce optimal input in certain data distribution. This forcing can be done by first Training on unsupervised mode is like adding prior

** → Problem would be That due To prior we would be unable To detect patterns That doesn't exist in unsupervised pretraining or was very far from it

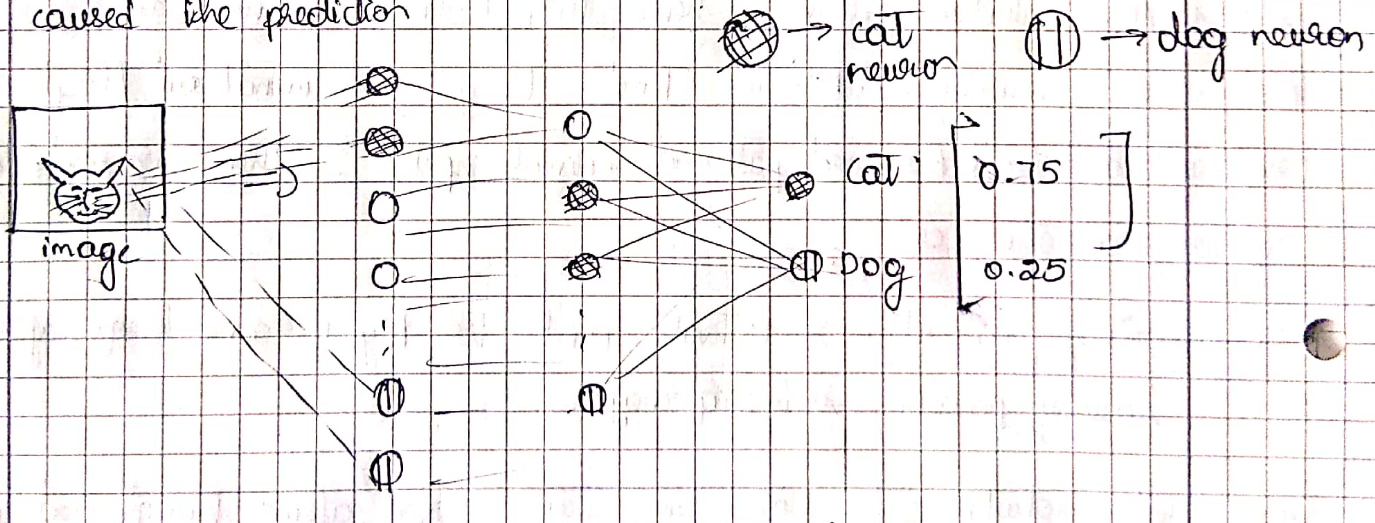
How To choose Them? There is no guideline, Need To Try Them based on our data & classifier

b) Attribution (Saliency Maps) : What in the input was important



→ Forward prop and find Feature maps.
→ Mask all values of feature map < 0 (ReLU mask) and find gradients and also consider only Top K pixels
→ Back propagate gradients only at Top K pixel position
→ This Tells how sensitive each input pixel is to output } Sensitivity analysis

Layer Wise relevance propagation : Where does info come from? What from input caused the prediction



→ Once we perform forward prop Take individual softmax and backpropagate layer wise (Do separately for Cat and Dog)
→ We can see That maximally activating neurons from Cat are due To "Whiskers" whilst for dogs neurons are focusing on "Ears"

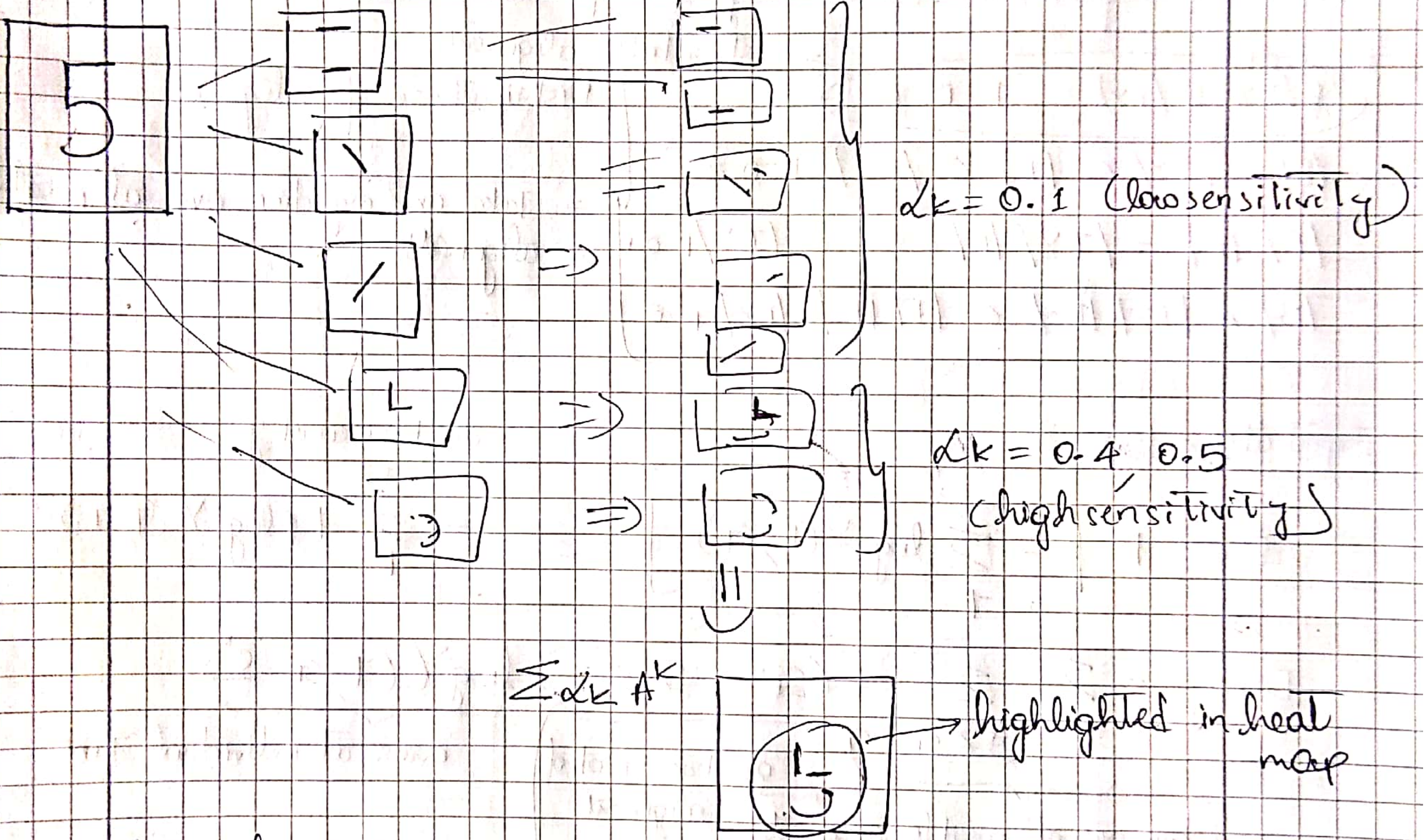
Grad CAM ÷ How sensitive is prediction To change in featuremap ?
complete

α_k^c → weighting on how sensitive featuremaps entirely rather Than individual position

$$L^c_{gradcam} = \mathrm{ReLU}\Big(\sum_k \alpha_k A^{k}\Big)$$

class → c ; k → Layer ; A → Forward pass activation

→ All feature maps are weighted and summed and put over eachother and produce heat maps



5
$\alpha_k = 0.1$ (low sensitivity)
$\alpha_k =$ 0.4 0.5 (high sensitivity)
$\sum \alpha_k A^k$
highlighted in heat map

Sanity checks for introspection

→ Most introspection models sucks and won't perform well for varied input. They don't make much sense and we must not Trust Them blindly

→ Less support for Segmentation Task as we need To apply for each pixel as each pixel can be segmented

→ Look at counterexample { Randomization of network destroys example ; Different classes have same explanation

✱ ✱

→ Most of Them cannot work with Batch norm and ReLU (+ve activation) and for real valued Input space (-ve i/p)