Cours

Charles Vin

Date

1 Feature detection & description

- Local detection/description, looking for invariance
- feature detection : Points/Regions of Interest detection : Corner detection
  - Detection of change in two directions with the eigenvalue of the Hessian matrix (one value = one direction)
  - Convolution with special filter $\rightarrow$ Large value = corner + comparaison with value before moving the windows $\rightarrow$ Threashold $\rightarrow$ corners
- Feature description :
  - Example : SIFT
  - Similar patch of image == close descriptor
- Bag of visual word of descriptor / features
  - Extraction (with feature descriptor or cnn) $\rightarrow$ into clustering (unsupervised learning ; K-Means, GMM, ...)
  -
- K-Means
  - Loss : $C(w) = \sum_{i=1}^{n} \min_k \|x_i - w_k\|^2$
  - pros : simplicty, convergence to local min
  - Cons : mémory intesive, choice of K, sensitive to init and artefacts, pherical clusters
- Image signature (not sure about this one) : matrix of Likelihood value, size $M \times K$ with K dico size, number of cluster extracted and $M =$ number of feature, then take the maximum likehood I think

2 SVM

- Problem : Donnée non linéaire $\rightarrow$ Projection, mais si projection dim ++ $\rightarrow$ Attention sur apprentissage + quel dim choisir $\rightarrow$ Solution : SMV do this auto
- Maximiser la marge $\gamma \Leftrightarrow$ minimiser $\|w\|$ sous la contrainte $\forall i, (wx^i + b)y^i \geq 1$ par des calculs obscures (≥ 1 car on veut que la distance entre la droite de régression et ces deux marges soit supérieur 1)
- Prise en compte des erreurs si pas de frontière linéaire pur : (soft margin)
  - ξ variable de débordement par rapport à sa marge pour chaque point mal classé $\rightarrow$ Raison obscure $\rightarrow \xi = \max(0, 1 - (wx^i + b)y^i)$ Hinge loss
  - On avait $\min ||w||^2$ maintenant $\min ||w||^2 + K \sum \xi$ avec K hyper param nombre d'erreur
  - $\|w\|^2$ = margin maximization, $K \sum \xi$ = Constraint satisfaction
  - $\|w\|^2$ = Régularization, $K \sum \xi$ = Data fitting
- The support vectors are the data points that lie on the margin, which is the region between the decision boundary and the closest data points of each class. Support vectors are critical in SVM because they determine the location and orientation of the decision boundary. All other data points that are not support vectors are not used to construct the decision boundary, which means that SVM is robust to noise and outliers in the data.
- La taille de la marge est un hyper-paramètre important : marge grande == underfitting // marge petite == overfitting (séparation linéaire plus proche des points, moins centrée)
- K petit = $K \sum \xi$ petit = petite pénalisation des erreur = tolérance de celle ci = underfitting // inverse
- Better on noisy problem

Kernel Tricks :

- Kernel Function : $k(x, y) =< \phi(x), \phi(y) >$
- Mesure la similarité entre 2 objets
  - – = vecteur opposé = éloigné
  - = 0 = produit orthogonal = éloigné
  - ++ = vecteur aligné = proche

3 CNN arch

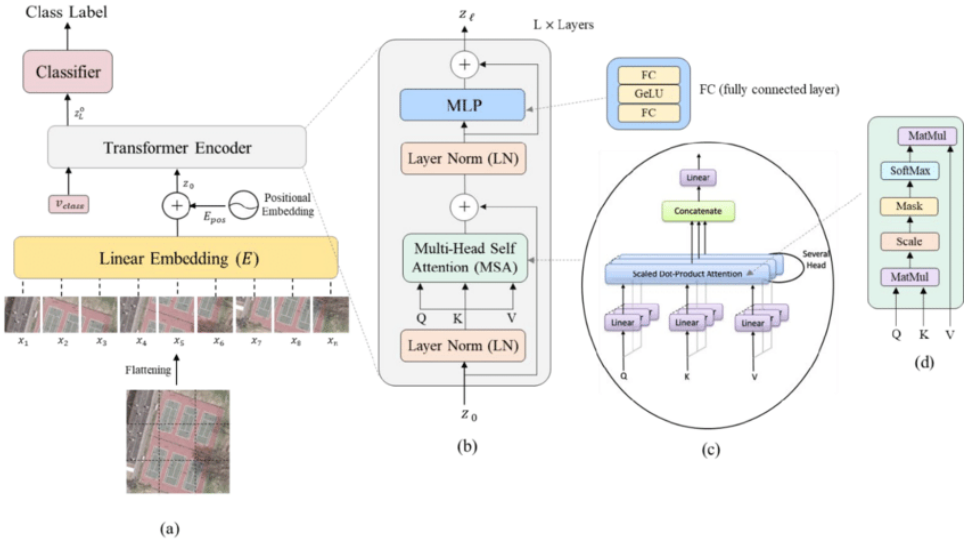
- LeNet5 : first CNN, for MNIST, FC at the end
- AlexNet : bigger, GPU, more data, dropout, pooling, relu, data aug, contrast normalization, FC at the end
- GoogLeNet : Inception layers, auxilary classifier + main classifier, less FC than alexnet at the end of each head (less params)
- VGG : rly deep, the deeper the better
- ResNet : good of deep, skip (residual) connection to stabilise training

4 FCN & Segmentation

FCN
- ImageNet : huge (1.2M), label, centered objects
- VOC, MS COCO : complexe scene : not centered, many different objects, variable size, background
- Large/Complexe images solution :
  - Resize : Naïve approach
  - Sliding windows
- J'arrive pas a capter le diapo, pourtant j'étais au cours ptdr, j'vais écrire ce que je comprend
- We're trying to make a network that output heat maps per class and can classify each image
- Heat map $\rightarrow$ score by class == not easy, need pooling ($h \times w \times K$ feature $\rightarrow h \times w \times C$ classes $\rightarrow$ $1 \times 1 \times C$ probabilities by classes)
- Different pooling method
  - GAP
  - Max
  - LSE
  - WELDON : max+min pooling : keep also the min to have a localized evidence of the *abscence* of the class. For example, take an image with a bed and a chair, the chair indicate a dinning room but the bed tells it's not !

Supervised Sementic Segmentation
- F-CN : upsample progressivly (pas clair help)
- DeepLab
  - Input : Heat map for one classes
  - $\rightarrow$ Upscaling with bi-interpolation and with atrous filtering (== conv with dilatation !)
  - $\rightarrow$ Fully-connected Conditionnal Random Field (CRF) :
    - Model conditional probability distributions
    - Graph of pixels
    - Use the intensity of the heat map pixel (unary term) and the relationships between pairs of neighboring pixels (pariwise term) i.e. penalizes similar pixels having different labels
    - $\rightarrow$ into an energy function $\rightarrow$ minimization of this function
- Deconvolution Networks
  - Mirrored version of the CNN
  - Unpooling layer : output = sparse activation map
    - Switch variable : where to had empty pixels (obtained during pooling (idk how))
  - Deconvolution : learning a filter to go from 1 pixel to 3×3
  - Add skip connection between the convolution and the deconvolution network $\rightarrow$ U-Net

5 ViT



— ConvNet = local attention (less local after many layers) VS ViT = global attention
— To achieve downstream task such as detection, segmentation, videos → Encore → process → decode, all with ViT. The decoder take a task specific query array.

6 GANs

— Auto encoder & VAE
  — problem : one pixel difference beweet generated image and the target can be either realistic or not
— Discriminator : generated image VS real images :
— Generator : random image → generation → discriminator
— Learning :
  — Discriminator : GD with a freeze generator
  — Generator : GD with a freeze discriminator
  — loop 50.000 times
— Math and Loss

$$V(G,D) = \mathbb{E}_{x\sim P_{\text{data}}}[\log D(x)] + \mathbb{E}_{x\sim P_G}[\log(1-D(x))]$$
$$G^* = \arg\min_G \max_D V(G,D)$$

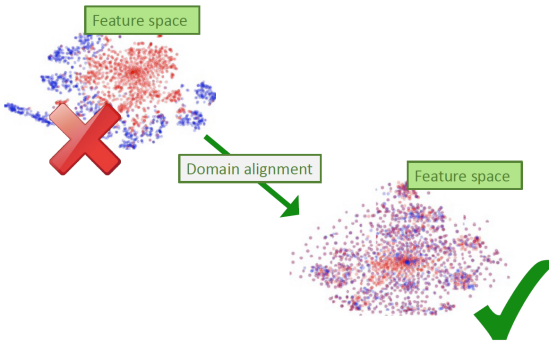
  — For the generator $\max_D V(G,D)$ evaluate the "difference" between P_G and P_{data}. The solution of this loss is $P_G = P_{\text{data}}$
  — $D(x)$ probability that $x \in P_G$ with $x \in P_{\text{data}}$ → to maximise
  — $D(G(z))$ probability that the output of the generator G is a real image → to maximise $\max_G D(G(z))$ $\Leftrightarrow$ to $\min_G 1 - D(G(z))$
  — D tries to maximize the probability it correctly classifies reals and fakes ($\log(D(x))$), and G tries to minimize the probability that D will predict its outputs are fake ($\log(1-D(G(z)))$).
— Evaluation :
— Cons : Learning can be hard : G and D must be well synchronized for convergence
— Pros : Computationnal efficient (no complexe likelihood inference), can fit sharper distribution, Spatial resolution, object quality
— Achitecture improvement
  — Laplacian Pyramid GAN (LAPGANs) (improve spacial resolution) : improve the generator : combines the strengths of Laplacian pyramids and GANs. It generates images in a coarse-to-fine fashion, where each level of the Laplacian pyramid refines the image details, leading to high-quality, high-resolution image generation.

- DCGANs (Improve object quality) : full conv generator and discriminator (no fcc, better activation fnc, batchnorm); upsampling step by step
- ProGANs : combine both idea : the network is trained incrementally, starting with low-resolution images and progressively increasing the resolution by adding layers to the network. This approach enhances the stability and efficiency of the training process, allowing the generation of high-resolution, detailed images with improved quality and variation. → complicated to train "out of the box" (block adding logistics and have to ajust params for each dataset)
- MSG-GANs : upsampling + skip connection between the generator and the discriminator for better learning
- StyleGANs : style transfer thing with control of the image generation process at different levels of detail through the use of adaptive instance normalization (AdaIN)

— Editing : Possible to change a generated image in the latent space to move it in a different class zone with linear interpolation (smiling womand - neutral women + neutral man = smiling man)

7 Conditional GAN

8 Domain adaptation



		Source Data (not direct labelled
Target Data	labelled	Fine-tuning *Multitask Learning*
	unlabeled	Domain adaptation-adversarial training ***Zero-shot learning***

Unsupervised Domain Adaptation (UDA)

— The core idea behind UDA is to align the feature distributions between the source and target domains.
— In the courses, we explored the Adversarial Training techniques, but there are also other
— Domain Adversarial Neural Networks (DANN)
  — Feature extractor →
    — → Label classifier : make predictions on the source domain data
    — → Domain Discriminator : classify the domain of the features (source or target)
  — Gradient Reversal Layer
    — The key innovation in DANN is the Gradient Reversal Layer, which is placed between the feature extractor and the domain discriminator.
    — During the backward pass, the Gradient Reversal Layer reverses the gradients flowing into the domain discriminator. This means that the feature extractor is encouraged to produce features that confuse the domain discriminator.
    — **Essentially**, the feature extractor is trained to make domain-specific information less distinguishable by the discriminator, encouraging it to learn domain-invariant features.
  — Loss = combines the classification loss from the label predictor (for the source domain) and the adversarial loss from the domain discriminator.
  — Once the DANN is trained, the shared feature extractor → extract features from the target domain → clasifier

Zero-shot

— Representing each class by its attributes → classify using a table of attribute by class

— Attribute embedding + class embedding $\rightarrow$ Take the nearest class? $\rightarrow$ Vision + language models
  — Triplet loss
    — 3 params : anchor point (the image), positive caption, negative captions
    — loss function aims to minimize the distance (or maximize the similarity) between the anchor and the positive example while simultaneously maximizing the distance (or minimizing the similarity) between the anchor and the negative example.

$$\text{Triplet Loss } = \max(0, margin + d(anchor, positive) - d(anchor, negative)).$$

where d(a, b) represents the distance or dissimilarity between data points a and b.