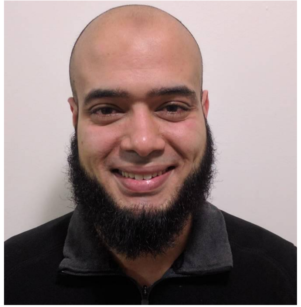
Machine Learning

Operator Invariance

Mostafa S. Ibrahim

Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / MSc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Notes

- In the RNN lectures, we discussed useful notes about invariance
- I later created slides for sake of knowledge saving

Operation Invariance

- Invariant: Imagine given a list of numbers
  - max([10, 20, 50) = max([50, 20, 10])
  - We say max function is permutation invariant function
  - This means, regardless of the order, it will give the same function
  - This means maxpool and average pool are permutation invariant operators
- Variant: Imagine given a list of numbers
  - Your function F(a, b, c) that computes a/b/c
  - This is permutation variant: F(100, 20, 10) != F(10, 100, 20)
- We have many operations, e.g. translation, rotation, etc
- An invariant operator is not affected by the property

Translation Invariant

- Imagine we have an image with a cat in the top left
- The same cat moved by photoshop to be in the bottom right
- Picking a pixel in the middle, does a learned filter behave differently?
  - No
- We say then 2D Conv operator is translation invariant
  - Hence you don’t need a lot of data with translated objects!

Scale Invariant

- Imagine a cat with 3 scales: 100x100 and 200x200 and 1000x1000
- **Convolutional networks** can also exhibit a degree of scale invariance, especially when combined with pooling layers. Pooling layers reduce the spatial size of the representation, making the network more tolerant to variations in the size of the input features or objects. While **convolutional layers** themselves are **not inherently scale invariant**, the combination of convolutional layers with pooling (e.g., max pooling) layers in a deep network architecture can help achieve this property to some degree.
- So
  - Conv 2D: not scale invariant
  - Conv network: partially scale invarant

Rotation Invariance

- C**onvolutional layers are not inherently rotation** invariant
  - Certain network designs can achieve a degree of rotation invariance.
- This means the layer deals with a cat and 180 degree rotated cat very differently!
- What is the implication of that?
- We need our final model to NOT be sensitive to rotations!
  - But the network is sensitive
- This is why we do rotation augmentation so that the network see many such examples, but we don’t do translation augmentation as we don’t need!

Permutation Invariant

- Assume a dataset of 1M images
- Imagine a binary classifier that takes 3 images
  - Return true if they are of similar properties (e.g. all fruits)
- What is the implication if the network is permutation invariant?
  - We don’t care the order of the images. Just use 1M images
- What is the implication if the network is permutation variant?
  - To make the model invariant, you have to augment with possible permutations. In max case, factorial 3 is 9, so use 9M images

Misc

- If the model is invariant to property X, then we don’t need augmentations!
- However, if the model is variant, we must augment for it to force the model being invariant for such data
- On another perspective, the reason we do data normalization is to avoid the need of many examples or a lot of augmentations

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."