CMSC 491/691: Computer Vision **Fall 2023**

Lecture 10: 2024-03-13 Image Transformations

Lecturer: Tejas Gokhale *Scribe: Duong Ta*

The notes below are an example of what I am expecting. They were taken from a random graduate class. They illustrate some uses of various LaTeX macros. Take a look at this and imitate.

10.1 Recap of Last Lecture

The previous lecture covered a Pytorch program that demonstrated various tensor operations, including arithmetic, concatenation, and in-place operations. We then progressed to implementing a Convolutional Neural Network (CNN) on the Fashion-MNIST dataset. After training the model for 10 epochs, it achieved an accuracy of 71% on this dataset.

10.2 Box Prediction Metrics

We started with an image containing a duck and a tree. We aim to draw a bounding box around each object - one for the tree and another for the duck. Once these predicted bounding boxes are generated, how can we evaluate their accuracy? We hypothesize that comparing the distance between corresponding corners of the predicted and ground truth bounding boxes could be an effective approach.

Ponder This 10.1 *We note that a predicted bounding box that falls outside the ground truth bounding box and a predicted bounding box that falls within the ground truth bounding box can result in the same distance metric. Are there any other metrics we should consider?*

Answer: Yes. there is a metric called Intersection over union or Jaccard similarity expressed as:

$$IoU(A, B) = \frac{|A \cap B|}{|A| \cup |B|}$$

The ground truth box and predicted box are perfect aligned when A = B.

Note 10.1 *We note that we need a method to select predicted boxes.*

Answer: We use the non-maximal suppression method

1. Keep only peeks in detected responses
2. Remove low-probability boxes and keep high-probability boxes
3. *Greedy algorithm*: Using IoU and threshold to keep selecting boxes with high score

Note 10.2 *We note that we need other metrics for classes.*

Answer: For each class, we plot Precision-Recall Curve to get mAP score.

10.3 OWL-ViT: Open-vocabulary object detection model

Ponder This 10.2 *We introduce the long-tailed distribution problem, where a few objects have a high frequency of occurrence, while numerous other objects have a relatively low frequency. In scenarios where the training data is limited and imbalanced, are there any alternative methods we should consider?*

Answer: Yes, there exists a methodology for transferring image-text models to open-vocabulary object detection tasks. This approach leverages a standard Vision Transformer architecture with minimal modifications, coupled with contrastive image-text pre-training and end-to-end detection fine-tuning.

10.4 Image Transformations

We talked about 2D linear transformations of grayscale images. Specifically, each point (x, y) becomes a new point (x', y') by a linear transformation matrix $\mathbf{M}$ expressed as $\begin{bmatrix} x' \\ y' \end{bmatrix} = M \begin{bmatrix} x \\ y \end{bmatrix}$.

We discussed 6 methods of 2D linear transformations (see Table 10.1)

Method	Matrix
Scale	$\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$
Flip across y	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$
Rotate	$\begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$
Flip across origin	$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
Shear	$\begin{bmatrix} 1 & a \\ b & 1 \end{bmatrix}$
Identity	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Table 10.1: 6 methods of 2D linear transformations

Ponder This 10.3 *The limitation of 2D linear transformations is that they cannot represent translations, as no matrix transformation corresponds to translation computations. Is there a solution to address this?*

Answer: We add another dimension to the matrix transformation. Specifically, each point $(x, y, 1)$ becomes a new point $(x', y', 1)$ by a linear transformation matrix $\mathbf{M}$ expressed as $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = M \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$.

In the context of 3D transformations, we introduced two complex transformations that result from composing multiple linear transformations. The first is the affine transformation, and the second is the projective transformation.

10.4.1 Affine transformation

Definition 10.1 *Affine transformation is a combination of 4-DOF (degree of freedom) and translation*

The transformation is represented as:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

We discussed some properties of the affine transformation

1. origin does not necessarily map to origin
2. lines map to lines
3. parallel lines map to parallel lines
4. ratios are preserved
5. compositions of affine transforms are also affine transform

10.4.2 Projective transformation

Definition 10.2 *Projective transformation is a combination of affine transforms and projective wrapping*

The transformation is represented as:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

We discussed some properties of the projective transformation

1. origin does not necessarily map to origin
2. lines map to lines
3. parallel lines do not necessarily map to parallel lines
4. ratios are not necessarily preserved
5. compositions of projective transforms are also projective transforms

Additionally, we explored the method to determine the transformation function given the original shape and its transformed counterpart. Specifically, we minimized the least squares error function, which can be expressed as:

$$E = \sum_i ||f(x_i; p) - x_i'||^2$$

We can vectorize this function and leverage libraries like numpy.linalg.lstsq to find the solution.

$$E = ||MX - X'||^2$$