IMAGE SIMILARITY using C++, PYTHON and ONNX

ABSTRACT

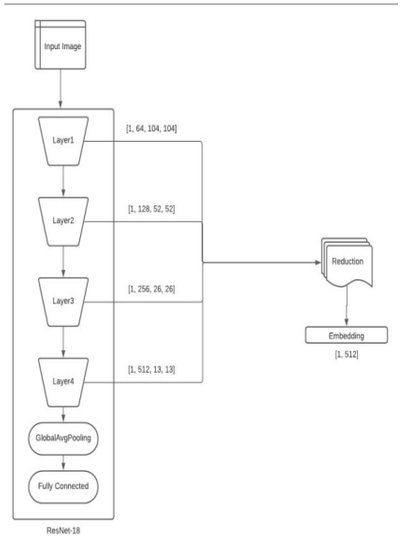
Nowdays, Customers are increasingly using social media platforms, such as instagram and snapchats, as a source of inspiration so the visual search has the potential to transform how we make easy to find relevent content or relevent product in term of shopping. Image similarity detection is used to quantify the degree of visual and sematic similarity of the images. Duplicate product detection , image clustering, visual search, and recommendation tasks are performed with this technology in modern applications.

INTRODUTION

An architecture of Image similarity is composed by three fundamental building blocks: a feature extraction (image vector), feature processing and similarity function. In this project our concern aims towards the latency of the project by which delay in process can be avoided. Representation of image in form of vector is been done by the already pre-trained models such as resent, xception, inception, alexnet etc. All the above model is already been trained on different image datasets according to their mean and standerd deviation by which image normlization takes place for convenient of computational complexity. Nowdays. Many similarity fuctoins is beeing used based on their application. Here, cosine similarity is been used. In terms of latency, hardware-level optimization is works better with c++ due it’s stong memory management, using CPU and GPU cores efficiently and compiler level optimization.

FEATURE EXTRACTION

In this porject, feature extraction has been done with using pretrained resnet-18 model. As from the latest competition of different neural net to perfrom task on images renet has perfomed very well as compare to others. Resnet-18 made of four layer(layer with pair of resblock). Extraction at each layer block has used for getting information as much as possible from image to understand the neighbour object or background understanding in image. Extraction at first resblock will give output 1*64*n*m, where n and m are hight and width of images and 1 and 64 are batch size and channel size respectively. Global avearage pooling applied for getting tensor in shape of 1*64*1*1 which is nothing but 1*64, similarly at each block extraction has applied with global average pooling in a way to get tensors in form of 1*64, 1*128, 1*256 and 1*256 respectively.



LATENCY

To emerge latency to pipeline, using custom operator written in c++ language to use it's strong memory management system, multitreading and compiler level optimization to make computatoin time faster as compare to python.

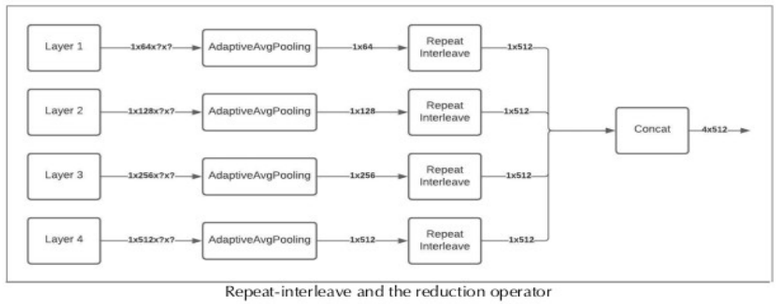
Example : execution time with using 1 thread took 508848 microseconds whereas multitread took 328890 microseconds.

[Mean accros all images for respective channels : { 0.0087, 0.0076, 0.0082 }]

[Standerd deviation accros images for channels : { 0.0038, 0.0040, 0.0024 }]

CUSTOM OPERATORS

Reduction & Repeat inteleave function written in c++ make computation fater and as compare to python. Repeat interleave hadles the size expansion of tensor to work with feature extracted at each resblocks. After the expansion of tensor at each resblock to 1*512 size tensor take average to hold information of each vector within main vector.



Repeat-interleave and the reduction operator

ONNX INTEGRATION

Onnx is an open formal built or a common file format which enables developers to use model with veriety of frameworks, tools, runtime and compilers. Here wrapper functions written in c++ language is being exported onto ONNX to by registering c++ operator with torchscript compiler.

SIMILARITY

Similariry of any image describes the common characteristics or distribution within image by which nature of object within image or common surrounding characteristics can be observed. Here, cosine similarity is used. Cosine similariy gives values between 0 to 1. In case of similarity value is being close to 1 states that both images are almost similar. In contrary, value being close to 0 or even near to 0.5 describes nature of dissimilarity between images.

$$\text{similarity} = \frac{x_1 \cdot x_2}{\max(\|x_1\|_2 \cdot \|x_2\|_2, \epsilon)}.$$

REFERENCES

ONNX operator tutorial :

https://github.com/onnx/tutorials/blob/master/PyTorchCustomOperator/README.md

Libtorch :

https://pytorch.org/tutorials/advanced/cpp_frontend.html