Object detection and computer vision
Assigment 3: Sketch recognition

Ugo Insalaco
ugo.insalaco@hotmail.fr

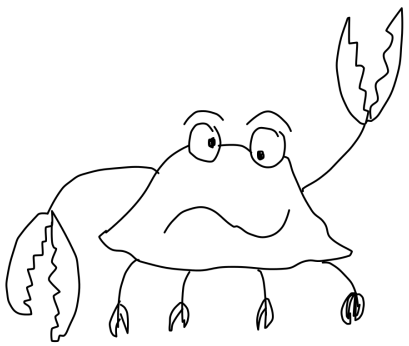

Figure 1. Example of a crab drawing from the classifysketch dataset

1. Introduction

During this assignment I implemented methods to classify hand-drawn sketches using machine learning methods on the classifysketch dataset [2]. I will focus on two methods, the first one using sift features and giving mediocre results and the second using neural networks producing easier and better results.

2. Sift extraction

2.1. Method

The code related to this experiment is located here. Following a method presented in the lessons and the previous assignments, I started by extracting a huge amount of features using the SIFT algorithm [4], thus creating a very large bag of visual words, and using a clustering algorithm (here Kmeans) I infer a fixed number K of representative features of the dataset which will constitute my dictionary. This dictionary is the foundation to preprocess the entire database, and format it in the shape of normalized histograms (using TF-IDF and the euclidian norm) which is the input data of a final SVM model.

dataset \ K	20	100	500	5000
train (acc. %)	18	60	70	70
val (acc. %)	6	10	11	6

Table 1. Results of the SIFT extraction for classification. The main issue is overfitting on training data

2.2. Results

The very critical parameter in this method is the dictionary size K. Having few reference visual words doesn't allow enough generalization power but a too large K would entice overfitting. In any case we found that this method was very much suffering from overfitting and very poor validation results compared to the training accuracy. Results are presented in table 2

3. Neural networks

3.1. methods

In order to learn from the sketch dataset, I also implemented and imported neural networks models using PyTorch.

Convolutional models: The first ones were two convolutional models using 3 or 4 convolution layers followed by max-pooling, batch norm and CeLu or ReLu activation functions. These models were call "basic CNN" and "basic CNN (larger)"

Fine-tuning models: Later I introduced ResNet18 and ResNet152 [3] as well as a transformer for image classification (vitB32 in [1]), pretrained on imageNet. These algorithms weren't at all good enough as zero shot methods so I fined tuned them on the sketch dataset.

Data augmentation: Many of the models were suffering from over-fitting. To address this issue I implemented some data augmentation methods in the pre-processing such as:

model \ dataset	train (acc. %)	val (acc. %)
basic CNN (larger)	10	7
ResNet18 (enh. 64 x 64)	85	56
ResNet50 (enh. 64 x 64)	90	66
ResNet152 (enh. 224 x 224)	86	70
vitB32 (base.)	0	0

Table 2. Results on training and validation sets of some of trained models. enh. : enhanced dataset, base. : base dataset

- Duplication of the dataset with horizontally flipped images
- Random Crop of the base image to a 758 x 758 image before resizing them to 64 x 64 or 224 x 224

Other addons: I also used the Adam optimizer to benefit from momentum at training.

3.2. results

Convolution models: During my experiments, I observed that adding the batch norm layers helped for the stability of the model (not collapsing to always predict the same value) and the CeLu activation to better learn. Although, being from the simplicity of the model or the lack of computing power, these models didn't manage to produce convincing results.

Finetuning models: From my experiments, ResNet was the easiest model to finetune reaching easily 40% accuracy at test time, however pushing the training it was able to master the training set while having poor results on the test set, which is why I needed data augmentation methods. For the transformer model I didn't manage to have it learn from the dataset, having its results stuck at less than a percent of accuracy.

4. Conclusion

The sift approach was not conclusive but at least allowed me to trained on some other old-fashioned image classification methods, and was the part on which I spent most of my time. The sad truth that I can conclude from my experiments is that adding more data and computing power was to me the only way to obtain a decent model: my last ResNet model was trained using the biggest ResNet152 and 224 x 224 dataset over a night. If I had more time I would have probably investigated more in using transformers as its poor results seemed odd. Another aspect that I would have tried, is having a better understanding of which object the dictionary of visual words represented, maybe selecting them manually, and using the histograms as input of a neural network and not a SVM as I did previously.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020. 1

[2] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31 (4):44:1–44:10, 2012. 1

[3] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015. 1

[4] Tony Lindeberg. *Scale Invariant Feature Transform*. 2012. 1