HAND-CRAFTED, GRADIENT-BASED METHODS COMPARED TO DEEP LEARNING WITH CONVOLUTIONAL NEURAL NETWORKS FOR ONE-SHOT FACE RECOGNITION

Kyle Maclean

July 2021

ABSTRACT

We present three methods for face recognition in a one-shot learning context where there is one example per class. The first method is a naive baseline. The others extract features from each training image to train one-vs-all SVMs [1], classifying each person as a class. For each testing image, the SVMs effectively determine which of the training images is most similar and assigns the training image's label to the testing image as the predicted class. The second method uses a HoG [2] feature vector to represent the images. Although it is more accurate than the baseline method, it is also slower due to the amount of processing required. The third method uses Transfer Learning to convert AlexNet [3] trained on ImageNet [4] into a specialised face feature extractor by retraining it on a large face dataset. The network learns the features which are most useful to recognise faces in different scenarios. The only processing required is to pass each training/testing image through 23 convolutional, pooling and ReLU layers. The third method is therefore very accurate and very efficient compared to the baseline. It achieves 79.68% accuracy in 13.64 seconds.

1. INTRODUCTION

All the methods compress pixel data into "feature vectors" / "features" which can be compared for similarity. Each method defines a feature extraction technique and a feature comparison technique. Feature extraction can be split into two kinds of approaches according to the pixels-to-feature-vector algorithm: "hand-crafted", where a human defines it; and "learned", where a human defines architectural components but the values of the parameters and importance of the operations are learned through experience. Our second method uses the hand-crafted approach and the third uses the learned approach. Hand-crafted features have been used since the early days of Computer Vision, when they were the only option. The theory behind them is well-understood and the techniques which use them have been carefully refined. The second method explores two particularly famous hand-crafted features, which are "gradient-based" because they construct features according to the relative intensities of pixels. We chose a gradient-based method due to its widespread historical use and significance. The first gradient-based feature is Histograms of Oriented Gradients (HoG), which has been shown to have high performance in human pedestrian detection, among others. It uses a sliding window across image cells to "find" rapid changes in pixel intensity, for example, at corners of objects. The second gradient-based feature is Speeded Up Robust Features (SURF) [5], a more efficient version of Scale Invariant Feature Transform (SIFT) [6], which was a breakthrough technique to detect many kinds of objects. It works by combining techniques like those used in HoG and provides invariance to scale, rotation, etc. so that unique and useful local regions in an image can be identified. Learning-based approaches, e.g., Convolutional Neural Networks (CNNs) can be used to either extract image features or "end-to-end" in which one network does the feature extraction in earlier layers and classification in later layers. We only discuss the former since preliminary experiments showed poor results for end-to-end CNNs, specifically with Siamese Networks. We hypothesise that CNNs trained on hundreds of thousands of images will extract more useful features than hand-crafted methods.

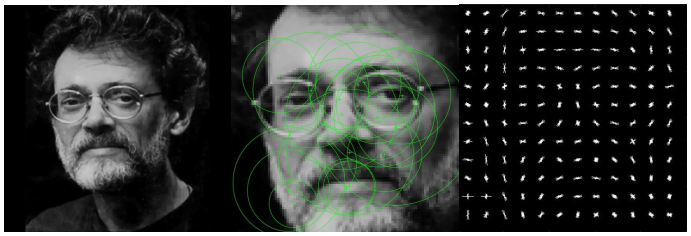
2. METHODOLOGY

2.1. Baseline method

Given an image, the baseline method normalises each pixel intensity into the range 0-1; calculates zero-normalisation of each pixel intensity by subtracting mean intensity and dividing by standard deviation of intensity; collapses the resulting matrix into a vector. For a testing image feature vector, it performs the dot product with each training image vector to obtain a cross correlation value, which is used as a similarity measure. The train/test pair with the highest cross correlation is deemed a match.

2.2. Gradient-based method

HoG uses an evenly-distributed grid of features derived from the whole input image of pixel intensities while SURF is "point-based" in that it returns unique image points, which may be unevenly distributed and have different locations, depending on the image. Since HoG describes the whole image and not points which may or may not be found between two different images of the same face (as in SURF), HoG is ex-



(a) (b) (c)

Fig. 1. Visualising the output of the two gradient-based methods. (a) Input image: 600*600*1px (grey). (b) The 20 strongest detected SURF (2.2.1) features displayed over the image after VJ cropping: the centres of the circles are on parts of the image that are useful in recognition. (c) Oriented gradient histograms (2.2.2) after VJ cropping; the cells are arranged in the same locations as their pixel constituents: the small lines mark major edges of the image (eyes, nose, mouth, face outline, etc.).

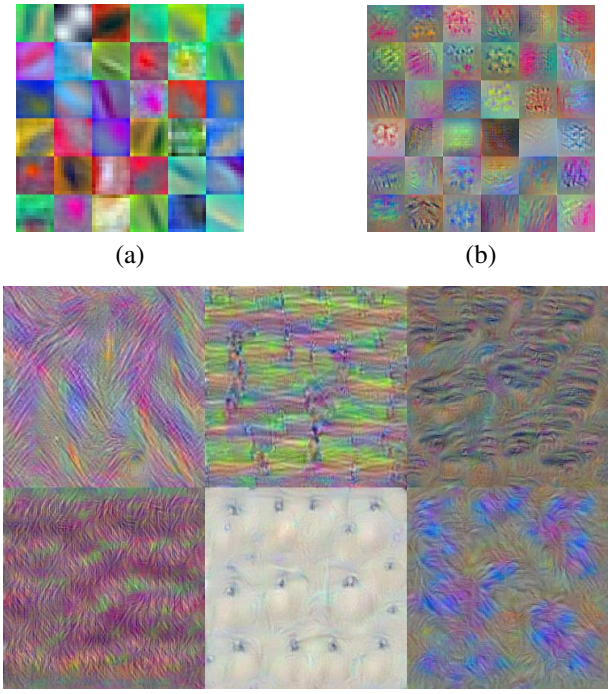
pected to perform better. Both feature types use the same image pre-processing pipeline: convert to grey-scale since they are only concerned with relative pixel intensity; crop to the extremities of a face which we detect using Viola-Jones [7]; down-sample to decrease computation time without losing significant information; train a one-vs-all SVM to determine the most similar training image feature vector for a given testing image feature vector to assign the corresponding label. Preliminary experiments with one-vs-one coding showed it to dramatically increase execution time without increasing accuracy, probably because it exhausts all combinations of class pair assignments while one-vs-all only exhausts all combinations of positive class assignments [8] and is thus more efficient.

2.2.1. SURF

Since there may be different numbers of points detected across different images, we restructure the set of SURF features into a Fisher Vector [9], where the length of the vector is constant. We tuned the number of clusters of the Gaussian Mixture Model [10] used to construct the visual word dictionary as a hyper parameter. This method requires the VLFeat library [11].

2.2.2. HoG

We tuned the Cell Size of the HoG grid as a hyper parameter. When increased, details smaller than the size of the cell may be lost but larger-scale information, like major face components, become more easily discernible [12].

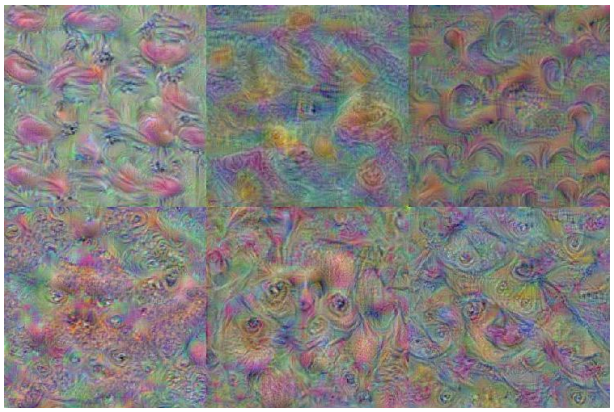


(a) (b) (c)

Fig. 2. Using MATLAB's `deepDreamImage` [13] function to visualise the learned filters of KyleNet's convolutional layers. The receptive field size increases after each layer, which is why we see lower-level features detected in the earlier layers and more complex patterns detected in later layers. (a) Convolution Layer 1 seems to pick up basic colours. (b) Convolution Layer 2 seems to have isolated shapes and textures. (c) Convolution Layer 5 seems to have a filter for different face components; we might see eyes in the bottom middle, ears in the top right, etc.

2.3. Deep Learning method

"Deep Learning" refers to Artificial Neural Networks with more than one hidden layer [14]. Almost all CNNs are deep because they use each hidden layer to perform one of a series of transformations; a typically a cycle of Convolution-Activation-Pooling. The coefficients of the convolution filter are learned through back-propagation while the Activation and Pooling layers are defined by the architecture designer. Its strength is that a well-designed architecture can have its convolutional layers learn features which are important for the problem (face recognition in our case) that humans may not have thought to use. Since it requires great expertise to design CNN architectures and lots of computation resources to train from scratch, we investigate pretrained networks, which have shown high performance for other problems [15]. After finding the best-performing pretrained network, we retrain it using the principle of Transfer Learning. It involves taking a network with weights that is broadly suitable for our task and:



(a)

(b)

Fig. 3. Using MATLAB's `deepDreamImage` function to generate images that resemble each class using KyleNet's final Fully Connected Layer. (a) Images that resemble the Fully Connected Layer from 6 selected classes: we might see impressions of entire faces here, especially in the bottom middle. (b) Random examples from the 6 selected classes: we might see resemblances between the distinctive parts of some of these faces and the corresponding generated image of their class, especially the ears in the top right.

1. replace its final layers to be compatible with our number of classes; 2. fine-tune its weights by training it on examples which are more relevant to our specific problem. This process is faster and requires less expertise than training from scratch. There are variations on how exactly to do Transfer Learning, and many of them have shown high performance [16].

2.3.1. Extracting features without Transfer Learning

MATLAB includes some famous CNN architectures which classify objects in the ImageNet database well [17]. High-performing weights for each architecture are also included, allowing us to use them as pretrained networks. We only use each network up to its Global Average Pooling layer because this contains the final feature vector representation of the image. The layers which come after it are usually: Fully Connected; Softmax; Final Output Layer (with class labels). They are all 1000-dimensional to classify the 1000 object classes in ImageNet and are therefore irrelevant to retrieving image features. For each experiment, we use a pretrained network to extract a feature vector whose dimensionality is defined according to the Global Average Pooling layer of the pretrained network. An SVM is then trained to learn the labels of each training image from these vectors. We tested both one-vs-one and one-vs-all SVM types.

2.3.2. Extracting features with Transfer Learning

We deemed AlexNet [3] to have the best balance between accuracy and efficiency out of all the tested pretrained CNNs (see Table 2). Now, we use Transfer Learning with a database of only face images to improve the usefulness of the extracted features. The CASIA WebFace dataset contains 494414 images of 10575 different people. It is no longer available on its official link, so we use an unofficial re-upload [18]. It was the biggest publicly-available face dataset we could find. The strength of using many different faces is that it forces the model to learn more universally-applicable features to recognise faces in general, instead of specialising to a small subset of the human population which a smaller dataset would provide. Our goal is not to classify CASIA images but to extract useful features for face recognition. The process of getting good validation scores on CASIA implies that our convolutional layers are better-tuned to extract useful face features.

The parameters for the Transfer Learning process are as follows: 70/30 train/validation split on the CASIA dataset; resize images to 227*227*3px (RGB) as required for AlexNet; replace layers 23 ("fc8") and 25 ("output") which are 1000-dimensional (for ImageNet) with 10575-dimensional (for CASIA) layers; train with the stochastic gradient descent with momentum (SGDM) optimiser with a tunable number of epochs; learn at a tunable rate; shuffle every epoch; and validate approximately once after seeing every 10% of the training files. For more information, see the file `TransferLearning.m`.

3. EVALUATION

Experiments were performed on MATLAB R2020b [19] running on a 6-core, 12-thread CPU at 3.6GHz [20] with 16GB RAM and a GPU at 1665 MHz with a 4GB frame buffer [21]. Each results table has a *Hyper parameters* column group which are varied to alter the method performance, as reflected in the *Performance* column group, defined by **Acc. (%)**: the Recognition Accuracy as a percentage of correctly labelled images, and **Time (s)**: Execution Time in seconds to process the training data and make predictions on the testing data. All tables are sorted in descending Recognition Accuracy order

and show only the 5 top-performing hyper parameter combinations.

3.1. Baseline results

The Baseline Recognition Accuracy is 25.37% and the Execution Time is 24.43 seconds.

3.2. Gradient-based results

Trying 20-100 clusters in the Fisher Encoding SURF solution's GMM only achieved accuracy between 24% and 26%. HoG is preferred because the best hyper parameter combination achieved 43.75% accuracy (see Table 1).

Hyper parameters			*Performance*	
HCS	**RIR**	**MRF**	**Acc. (%)**	**Time (s)**
8	100	200	43.75	41.51
4	100	200	42.56	55.05
8	100	300	42.56	42.73
4	100	300	42.49	54.85
4	50	200	42.49	41.25

Table 1. Performance of the HoG method: HCS = HoG Cell Size; RIR = Resized Image Resolution; MRF = Minimum Resolution in which to detect a Face.

3.3. Deep Learning results

From the results in Table 2, we chose AlexNet as the best performance compromise. It is especially valued for its efficiency (5.48 seconds to execute) because Transfer Learning will increase its accuracy but not make it faster. We use the activations from the layer called "drop7" as 4096-dimensional feature vectors for each image because it is the last layer before the 1000-dimensional layers. After Transfer Learning, the Execution Time roughly doubled, probably due to the time it takes to load in the custom model which has more than 10x as many classes as the original AlexNet (this is another reason to prefer the quicker AlexNet over more accurate, slower models). We tried to load only up to the Global Average Pooling layer to save time but this does not seem to be possible in MATLAB. Accuracy dramatically increased to 71.94% using the default learning rate and number of epochs. We increased accuracy further by quadrupling the number of epochs and diving the Initial Learning Rate by four. This prevents the network from adjusting weights too fast in response to incorrect classifications during back propagation and does not introduce any more overfitting because the total possible learning magnitude is unchanged. The final network is dubbed: KyleNet. It is 79.68% accurate and takes 13.64 seconds to run. It seems anomalous that the Execution Time is different when multiplying ME and dividing ILR by some number compared to not. In theory, having trained for more/less time should not change how long it takes to send input through the network during testing.

Hyper parameters		*Performance*	
Architecture	**SVM type**	**Acc. (%)**	**Time (s)**
densenet201	onevsall	37.20	29.50
alexnet	onevsall	37.12	5.48
efficientnetb0	onevsall	35.78	13.95
resnet101	onevsall	35.19	16.29
googlenet	onevsall	34.89	6.34

Table 2. Performance of the features extracted by different pretrained networks *without* transfer learning. Note that we also analysed the `onevsone` SVM type for each Architecture but none featured in the top 5.

Hyper parameters		*Performance*	
ILR	**ME**	**Acc. (%)**	**Time (s)**
75e-6	44	79.68	13.64
14.58	50	78.19	14.58
1e-4	33	77.97	12.96
15e-5	22	75.59	10.13
3e-4	11	71.94	10.82

Table 3. Performance of the features extracted by AlexNet *with* transfer learning after training on 10575 face classes. ILR = Initial Learning Rate; ME = Maximum Epochs. We experimented up to (Me=40; ILR=3e-4), and also using a 1000 class subset with the most examples, but none of these featured in the top 5.

4. CONCLUSION

Passing images through a CNN is computationally efficient compared to the processing required by Gradient-based methods. CNNs are demonstrably faster by an order of magnitude. This makes them suitable for real-world deployment because they can be used effectively on commodity hardware. However, 300MB is required to store our CNN. When memory is at a premium, methods consisting of hand-written computer code (typically requiring less than 10MB of space) may be preferred, especially for simpler problems like digit recognition. Hand-crafted methods are generally more interpretatable compared to the "black box" nature of ANNs in general, but this is not particularly relevant to CNNs, where we know what each layer does and are able to visualise their learned filters, as shown in Figures 2 and 3. Future research could investigate more complex network architectures (Table 2) for Transfer Learning. Execution Time and Accuracy may increase, which may be desirable when time is less of a constraint.

5. REFERENCES

[1] A. Tzotsos and D. Argialas, "Support Vector Machine Classification for Object-Based Image Analysis," in *Object-Based Image Analysis: Spatial Concepts for Knowledge-Driven Remote Sensing Applications*, ser. Lecture Notes in Geoinformation and Cartography, T. Blaschke, S. Lang, and G. J. Hay, Eds. Berlin, Heidelberg: Springer, 2008, pp. 663–677.

[2] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. San Diego, CA, USA: IEEE, 2005, pp. 886–893. [Online]. Available: http://ieeexplore.ieee.org/document/1467360/

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: https://dl.acm.org/doi/10.1145/3065386

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[5] G. Du, F. Su, and A. Cai, "Face recognition using SURF features," in *Sixth International Symposium on Multispectral Image Processing and Pattern Recognition*, M. Ding, B. Bhanu, F. M. Wahl, and J. Roberts, Eds., Yichang, China, Oct. 2009, p. 749628.

[6] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. [Online]. Available: link.springer.com/10.1023/B:VISI.0000029664.99615.94

[7] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1. Kauai, HI, USA: IEEE Comput. Soc, 2001, pp. I–511–I–518. [Online]. Available: http://ieeexplore.ieee.org/document/990517/

[8] "Fit multiclass models for support vector machines or other classifiers - MATLAB fitcecoc - MathWorks United Kingdom." [Online]. Available: uk.mathworks.com/help/stats/fitcecoc.html

[9] Z. Boulkenafet, J. Komulainen, and A. Hadid, "Face antispoofing using speeded-up robust features and fisher vector encoding," *IEEE Signal Processing Letters*, vol. 24, no. 2, pp. 141–145, 2017.

[10] B. Klein, G. Lev, G. Sadeh, and L. Wolf, "Fisher vectors derived from hybrid gaussian-laplacian mixture models for image annotation," *CoRR*, vol. abs/1411.7399, 2014. [Online]. Available: http://arxiv.org/abs/1411.7399

[11] "VLFeat Tutorials: Gaussian Mixture Models." [Online]. Available: https://www.vlfeat.org/overview/gmm.html

[12] "Extract histogram of oriented gradients (HOG) features - MATLAB extractHOGFeatures - MathWorks United Kingdom." [Online]. Available: uk.mathworks.com/help/vision/ref/extracthogfeatures.html

[13] "Visualize network features using deep dream." [Online]. Available: https://mathworks.com/help/deeplearning/ref/deepdreamimage.html

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: https://www.nature.com/articles/nature14539

[15] D. Marmanis, M. Datcu, T. Esch, and U. Stilla, "Deep learning earth observation classification using imagenet pretrained networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 1, pp. 105–109, 2016.

[16] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun, "A practical transfer learning algorithm for face verification," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[18] "CASIA-Webface dataset download link · Issue #18 · happynear/AMSoftmax." [Online]. Available: https://github.com/happynear/AMSoftmax/issues/18

[19] "MATLAB R2020b." [Online]. Available: https://mathworks.com/products/new_products/release2020b.html

[20] "AMD Ryzen™ 5 3600 Desktop Processor." [Online]. Available: https://www.amd.com/en/products/cpu/amd-ryzen-5-3600

[21] "GeForce 1650 Graphics Card." [Online]. Available: https://www.nvidia.com/en-gb/geforce/graphics-cards/gtx-1650/