Postprint: Optimizing Convolutional Neural Network-Based Image Object Classification Algorithm Using Adaptive False Nearest Neighbors Method

Authors: Huang Zhen, Li Zhihao, Yuan Yi, Ruan Wenhui

Date: 2018-08-13T00:00:00+00:00

Abstract

To achieve favorable results when processing diverse data types or tasks, a convolutional neural network (CNN) architecture based on the adaptive false nearest neighbors method is designed. The concept of central moments is applied to the pooling operations of CNNs, the sparse filtering algorithm is utilized to enable unsupervised training, and the sizes of convolution masks (convolution kernels) and the numbers of convolution units (CNN neurons) per layer are configured; furthermore, this architecture employs the adaptive false nearest neighbors method to accomplish tasks such as simplified modeling and prediction. Experimental results demonstrate that the proposed improved CNN architecture exhibits lower complexity, can be trained more rapidly, and is less susceptible to overfitting.

Full Text

Preamble

Image Target Classification Algorithm Based on Adaptive False Nearest Neighbor Method for Convolutional Neural Network Optimization

Huang Zhen[1] , *Li Zhihao*[2], *Yuan Yi*[1] , *Ruan Wenhui*[1]

1. Lanzhou University of Arts and Sciences
   a. School of Digital Media; b. School of Media Engineering, Lanzhou 730000, China

2. School of Information Science & Engineering, Lanzhou University, Lanzhou 730000, China

Abstract: To achieve good results when dealing with different data types or tasks, this paper designs a convolutional neural network (CNN) architecture based on an adaptive false nearest neighbors method. The approach applies central moment concepts to CNN pooling operations, utilizes sparse filtering algorithms to enable unsupervised training, and determines the size of convolution masks (kernels) and the number of convolution units (CNN neurons) per layer. Additionally, the architecture employs an adaptive false nearest neighbors method to simplify tasks such as modeling and prediction. Experimental results confirm that the proposed improved CNN architecture has lower complexity, can be trained more quickly, and is less prone to overfitting.

Keywords: convolutional neural network; false nearest neighbors; target classification; moment pooling; sparse filtering

0 Introduction

Deep learning (DL) algorithms have become a primary method for analyzing massive datasets, capable of extracting meaningful features to help understand and solve complex big data processing problems. DL algorithms operate across multiple hierarchical levels, each composed of regression models involving linear and nonlinear components. The combination of multiple models enables representation of complex functions. Most DL algorithms resemble artificial neural networks, where input vectors are processed through successive layers of operational units that emphasize or suppress features. Currently, DL algorithms are mainly applied to data classification [**?**], image recognition [**?**], and machine translation [**?**].

Convolutional neural networks (CNN) [**?**] are a classic DL algorithm that can extract specific features from time-correlated data such as audio and video. However, CNN performance depends heavily on its architecture, including the number of layers, units per layer, and convolution mask sizes. Moreover, architectures suitable for specific problems may not generalize when processing different data types or tasks. A common alternative approach involves evaluating different CNN architectures to select the best-performing one, which has several obvious drawbacks: (a) training and evaluating a single architecture is already computationally intensive, making comprehensive evaluation of many scenarios potentially infeasible; (b) empirically defined architectures may be unsuitable for the problem at hand and thus may not yield ideal results.

Strategies for avoiding evaluation of multiple CNN settings typically require additional training phases to adjust weights associated with convolution units, which increases computational burden and demands thousands or millions of examples to produce reasonable results [**?**]. Another strategy uses CNN ensembles, which typically allow leveraging maximum iteration counts but require more architectures to obtain relevant results.

This paper proposes a novel CNN architecture that applies central moment concepts to pooling operations and uses sparse filtering algorithms to achieve unsupervised training. Inspired by the False Nearest Neighbors (FNN) method [**?**], our approach analyzes input and output images produced by convolution operations at each CNN layer to estimate appropriate convolution mask sizes and the number of convolution units per layer, employing an adaptive false nearest neighbors method to simplify modeling and prediction tasks.

---

1 Related Research

Equation (1) defines the convolution operation, where $\mathbf{I}$ is input data (typically an input image) of size $p \times q$, is a convolution mask (or kernel) of size $m \times n$, and $\phi_{i,j}$ is the value produced by region convolution.

The convolution step defined in (1) can be written as a matrix-vector product in an mn-dimensional space. Let $\mathbf{v}_{i,j}$ be the vector corresponding to the local region of $\mathbf{I}$, and let be the vector representing the convolution mask. The convolution result is added to a bias vector $\mathbf{b}$. The maximum value in $\mathbf{z}$ becomes the output. This model achieves an error rate of 0.45%.

Assuming a convolution layer with l units (or neurons), where each unit corresponds to an $m \times n$ convolution mask $_k$, $k = 1, \ldots, l$, and using the previously described vector representation with a fixed local region $\mathbf{v}_{i,j}$ of $\mathbf{I}$, we can write each convolution step with $_k$ as the following matrix-vector product:

In other words, the convolution step corresponds to a linear transformation from $\mathbb{R}^{mn}$ to $\mathbb{R}^l$ to obtain the vector $\mathbf{z}$. However, MENNDL and all PSO methods also train each architecture to compute the fitness function until good parameters are found, which undoubtedly increases computational load.

CNN algorithms are widely applied to classification problems. Reference [**?**] uses Maxout models and dropout strategies to improve CNN feature extraction. Maxout is an activation function that multiplies convolution operation results by a diagonal weight matrix $\mathbf{W}$ and selects the maximum output. Reference [**?**] proposes a multi-scale CNN (MS-CNN) classifier with two-level features. Reference [**?**] studies sparse networks (SNoW) for image edge representation and object identification from the Columbia University Image Library (COIL-100). Each unit in SNoW represents a subnetwork responsible for classifying objects into specific classes, thus SNoW contains as many units as labels.

Reference [**?**] proposes a new framework that reduces training time and uses deconvolutional networks to evaluate training behavior. Correlation analysis measures similarity between input and reconstructed images as a fitness function, applying the Nelder-Mead method to optimize this function and find optimal structures. The framework provides parameter ranges and sufficient accuracy for analyzed datasets. Reference [**?**] uses multi-node evolutionary neural networks

for deep learning (MENNDL) to find optimal convolution mask sizes and filter numbers.

Our goal is to obtain simple CNN architectures for classification tasks and validate them using datasets. Our method demonstrates performance similar to deep networks in many experimental scenarios. Simple architectures enable lower processing costs and reduce overfitting.

2.1 Convolutional Neural Networks

CNN is a deep learning algorithm for processing multidimensional data such as signals, images, and video, capable of extracting relevant features even in the presence of noise, shift, and scaling. CNN is a multilayer network where each layer consists of units performing different operations, with convolution used to identify local recurrent features.

A typical CNN architecture has the following sequential structure: (a) the first layer represents input data without performing operations (serving as a validation function); (b) the second layer computes convolution operations with activation functions; (c) the third layer performs subsampling through pooling operations; layers (b) and (c) can be repeated; (d) the final layer produces output features.

2.2 Moment Pooling

In recent years, moment features have been increasingly applied in computer vision, pattern recognition, and image classification. Reference [**?**] proposes using moment features to estimate principal directions when extracting binary features to improve robustness. This paper applies central moment concepts to CNN pooling while using first-order matrices to reduce CNN model complexity.

The grayscale matrix is represented as:

$$\mathbf{I} = \begin{bmatrix} I_{00} & I_{01} \\ I_{10} & I_{11} \end{bmatrix}$$

where I_{xy} represents the grayscale value at image coordinates (x, y).

The central moment matrix is:

$$\mathbf{c}_{xy} = \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix}$$

From these two equations, we obtain the central moment of the pooling region:

$$\phi_{i,j} = \sum_{x=-m/2}^{m/2} \sum_{y=-n/2}^{n/2} (x - \bar{x})^p (y - \bar{y})^q I_{i+x,j+y}$$

This is a floating-point value that does not point to a specific discrete value. As shown in [Figure 1: see original paper], the fixed points representing the upper and lower boundaries of $\phi_{i,j}$ have fixed positions and can be called the four-neighborhood of $\phi_{i,j}$. We use interpolation to calculate its response value.

Common interpolation methods include nearest-neighbor interpolation, which is fast but can cause checkerboard effects, and bilinear interpolation, which performs linear interpolation along horizontal and vertical axes, weighting distances to the insertion point. This algorithm may blur contour information and involves higher computational complexity.

Based on probability p_{xy}, we randomly select (x, y) on the horizontal and vertical axes. The closer to the central moment, the higher the selection probability, as shown in equations (7) and (8). We then select the response value of the pooling region from the four-neighborhood according to the value of p_{xy}. Random selection can avoid overfitting and has low computational complexity.

2.3 Unsupervised Training

A challenge in CNN applications is obtaining large amounts of labeled training data, which unsupervised training can address. Using sparse filtering algorithms, we construct a feature distribution matrix and apply normalization to process sample features, obtaining a sample distribution matrix with sparsity in both sample distribution and activation time. Furthermore, cascading sample distribution matrices can form unsupervised learning models.

Let $\mathbf{X} \in \mathbb{R}^{t \times n}$ be the constructed feature distribution matrix, where each row represents a feature and each column represents a projected training sample. Let $\mathbf{W} \in \mathbb{R}^{d \times t}$ be the projection matrix. The sparse filtering algorithm aims to find the optimal solution for $\mathbf{W}$ that satisfies three mentioned properties.

Let $\mathbf{f}_i$ be the i-th row of $\mathbf{F}$ and $\mathbf{f}_j$ be the j-th column of $\mathbf{F}$. We perform ℓ_2 normalization on rows and columns of $\mathbf{F}$:

$$\tilde{f}_i = \frac{f_i}{\|f_i\|_2}, \quad \tilde{f}_j = \frac{f_j}{\|f_j\|_2}$$

By optimizing $\mathbf{F}$ for sparsity, we obtain the projection matrix $\mathbf{W}$:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_{i=1}^{d} \|\tilde{\mathbf{f}}_i\|_1$$

Cascading the sample distribution matrix and treating $\mathbf{W}$ as convolution kernels yields an unsupervised learning model. This structural model uses a greedy algorithm for layer-by-layer solving during training, where the next layer' s optimization only begins after the current layer' s optimization is complete.

2.4 False Nearest Neighbors

The False Nearest Neighbors (FNN) method is a technique for unfolding time-correlated data into multidimensional space (phase space) to simplify modeling and prediction tasks. It iteratively increases phase space dimensionality, measuring changes in neighborhood relationships at each step. FNN first reconstructs the original dataset into an m-dimensional space (typically $m = 2$) with fixed time delay τ (typically $\tau = 1$), where the i-th nearest neighbor of state $\mathbf{x}(m)$ is $\mathbf{x}^r(m)$. After adding a new dimension, FNN calculates the distance change of the nearest state $\mathbf{x}^i(m+1)$ according to equation (12). If $V_{i,m+1} > V_{th}$, the state is considered a false neighbor, where V_{th} is a user-defined threshold.

When data contains noise or insufficient samples, this criterion may compute erroneous estimates. This can be avoided using equation (13), where $D_{i,m}$ represents the distance between neighbors, $\bar{D}$ is the average distance between each point and all other points, and A is a user-defined threshold.

Finally, FNN computes the fraction of false neighbors across the entire dataset to provide a cutoff point for defining the embedding dimension d.

2.5 Adaptive False Nearest Neighbors Method

To correctly estimate the convolution mask size for a given CNN layer, we adapt the False Nearest Neighbors (FNN) method. Considering two embedding dimensions: one applied along rows (M) and another along columns (N), they together define vector $\mathbf{v}_{i,j}$.

Let $\mathbf{I}$ be the training sample matrix. We apply convolution on vector $\mathbf{v}_{i,j}$ within a window to obtain a new space $\mathbf{S} = {}^{T}$, where represents the convolution mask. The FNN method analyzes the vector set after phase space reconstruction.

Algorithm 1: Analyzing False Nearest Neighbor Values of Vector Set After Phase Space Reconstruction

Data: $\mathbf{v}_{i,j}$ - vector set from given data $\mathbf{I}$; nn - number of nearest neighbors; at - threshold radius for neighborhoods; τ - vector bias obtained from $\mathbf{v}_{i,j}$.

Result: ff - false neighbor values.

1. Initialize matrix $\mathbf{F}$ with $\mathbf{v}_{i,j}$
2. Initialize counter $= 0$ and denom $= 0$
3. For each vector $\mathbf{v}_{i,j}$ in $\mathbf{F}$:
    a. Find the nearest vector $\mathbf{v}_{idx}$ in $\mathbf{F}$
    b. Compute Euclidean distance $d_{i,idx} = \|\mathbf{v}_{i,j} - \mathbf{v}_{idx}\|$
    c. For each neighbor $\mathbf{v}_{idx}$ in $\mathbf{F}$:
        i. Compute $d^{\tau}_{i,idx} = \|\mathbf{v}_{i,j+\tau} - \mathbf{v}_{idx+\tau}\|$
        ii. If $d^{\tau}_{i,idx}/d_{i,idx} > at$, then counter ++
        iii. denom ++

4. ff = counter/denom

We also employ the same method to determine the number of convolution units per CNN layer. Convolving a region $\mathbf{v}_{i,j}$ of $\mathbf{I}$ with all unit masks maps $\mathbf{v}_{i,j}$ to a one-dimensional space where l is the number of convolution units. When $l < MN$, we can perform an inverse transformation to simplify training equations. When $l > MN$, dimensionality reduction eventually maps different vectors in the input space to the same vector in the output space. Therefore, we should seek $l \approx MN$.

Algorithm 2: Estimating Mask Size in Convolution Units

Data: Input data $\mathbf{I}$; $maxM$, $maxN$ - maximum rows and columns.

Result: $\mathbf{F}$ - matrix containing false neighbor values for all evaluated mask sizes.

1. Initialize matrix $\mathbf{F}$ with size $maxM \times maxN$
2. For $M = 1$ to $maxM$: For $N = 1$ to $maxN$:
   a. Embed input data $\mathbf{I}$ into multidimensional "phase space" with dimensions $M \times N$
   b. Store FNN results in $\mathbf{F}[M, N]$

Algorithm 3: Estimating Number of Convolution Units in a Given CNN Layer

Data: Input data $\mathbf{I}$; M, N - rows and columns per convolution mask; nn - number of nearest neighbors; at - threshold radius; τ - vector bias; $maxU$ - maximum number of units to consider.

Result: $\mathbf{U}$ - vector containing false neighbor values for each added convolution unit.

1. Initialize matrix $\mathbf{S}$ with $\mathbf{I}$
2. Initialize matrix $\mathbf{U}$ as empty
3. For $k = 1$ to $maxU$:
   a. Apply convolution on vectors $\mathbf{v}_{i,j}$ in $\mathbf{S}$ with k random convolution masks to obtain new space $\mathbf{S}_k = {}^{T}_{k}{}_{k}$
   b. Compute $\mathbf{U}[k] = \text{FNNAnalysis}(\mathbf{S}_k, nn, at, \tau)$

When adding convolution units, the number of false neighbors is expected to decrease. We estimate the number of units per CNN layer as the index of vector $\mathbf{U}$ that first reaches zero (or near-zero) for FNN measurement, defining the most appropriate dimensionality.

3.1 Datasets

We conduct experiments using three datasets, all split into training and test sets. The training set is used to estimate parameters via our FNN method,

which does not require label information. [Figure 2: see original paper] shows sample images from the datasets.

a) **CMU Face Images Dataset** [**?**]: Contains 1,872 grayscale face images with four variations: pose (left, right, up, down), expression (happy, sad, angry, neutral), eyes (with/without sunglasses), and size ($128\$\times 120, 64\times 60, and 32\times\$30$). We experiment with the smallest images (640 examples) and two different classes, splitting the dataset into 70% training and 30% testing.

b) **MNIST Dataset** [**?**]: A handwritten digit binary image dataset with 60,000 training samples and 10,000 test samples.

c) **COIL-100** [**?**]: The Columbia University Image Library contains 100 classes with 7,200 RGB images (355 images per class with pose variations). We resize images to $32\$\times\$32$ pixels for faster experimentation, using 70% for training and 30% for testing.

3.2 Setup

Adaptive FNN parameters are set as follows: number of nearest neighbors $nn = 10$; neighborhood radius threshold $at = 1$; time delay $\tau = 1$ for Algorithms 1-3. For Algorithm 2, maximum rows and columns are set to half the image size, so $maxM$ and $maxN$ are determined based on the optimal mask size for the analyzed dataset. Algorithm 3' s maximum convolution units $maxU = 50$.

SGD method is used to train all Caffe-based CNNs. We use successive layers of convolution, ReLU, and max-pooling. Max-pooling is applied after every $3\$\times 3 pixel ReLU layer with a stride of 2\times\$2$. Training parameters are: learning rate 0.001 (fixed, no decay), momentum 0.9, batch size 100 samples. Solver parameters are selected empirically to evaluate the impact of convolution mask size and units per layer.

3.3 Parameter Evaluation

We execute Algorithm 2 on a single image from the training set to provide FNN measurement matrix **F** while varying mask row and column sizes. We compute the average of these matrices when calculating FNN metric values, then use the norm of this average matrix to formulate a stopping criterion. If the difference between current and previous criteria is less than a user-defined threshold (set to 0.001), execution stops. Finally, we compute a histogram from the average matrix **F** to select the optimal convolution mask size. For each row in the matrix, we count columns presenting the first zero or minimum nearest neighbor score. We then perform the same along columns. The histogram is represented by a

matrix indicating which row and column combination is optimal, with maximum value 2 and minimum 0.

Considering the optimal mask size for each dataset, we run Algorithm 3 on each input image in the given training set to provide FNN measurement vector **U** while varying convolution units in a single layer. The unit count is given by the index of vector **U** containing the first zero (or lowest possible value), produced by averaging FNN measurements. This zero indicates the target space of the linear transformation is sufficient to represent the input space.

We use the MNIST dataset to compute results for our method in estimating mask size and units per layer. For mask size estimation, a random image from the training set serves as input until the stopping criterion is met (difference between current and previous average matrix norms falls below threshold). [Figure 3: see original paper] shows a training image and its corresponding FNN matrix, where (a) represents a training sample and (b) shows the false nearest neighbor score matrix for that sample, with the x-axis representing mask size by column and y-axis representing mask rows. We compute an element-wise average matrix from FNN matrices, then create a histogram on this final matrix to return the optimal convolution mask size.

shows the average vector for unit number estimation, also obtained from the MNIST dataset, considering a single convolution layer with mask size 8$×$10. The selected unit count is the index of the first zero in such a vector.

lists error rates (ER) for several selected convolution mask sizes and their respective unit counts, executed with 10,000 iterations in this deep learning framework to select the best convolution mask size.

3.4 Using Optimal Mask Size and Convolution Unit Numbers

After investigating optimal mask sizes and convolution unit numbers, we train our CNN architecture with optimal parameters for 50,000 iterations. presents error rates, corresponding mask sizes, and convolution unit numbers selected by our FNN method for all datasets. The results demonstrate that our FNN method provides good parameters even when evaluating mask sizes and unit numbers using only a small fraction of instances.

For the CMU Face Images dataset, we classify people in images. Our method achieves a minimum error rate of 0.56% with a single convolution layer. Our approach helps find the optimal linear transformation to provide relevant features based on vector neighborhoods in the shared domain. Results on this dataset show our strategy is useful for configuring required CNN architectures.

In reference [**?**], the MNIST dataset achieved a minimum error rate of 0.23%. With a single convolution layer, our method achieves 1.85% error, 1.88% with

two layers, and 3.02% with three layers. This is comparatively poorer because our experiments use simpler CNN architectures, while [**?**] employed a very complex structure containing 35 CNNs (MCDNN) with: 20 units in the first convolution layer, 20 units in the next subsampling layer, 40 units in the second convolution layer, 40 units in the next subsampling layer, 150 units in the MLP hidden layer, and 10 units in another MLP hidden layer.

Reference [**?**] used 100 deep convolutional wavelet networks (DCWN) to achieve an error rate of 0.8% on COIL-100, while our method achieves 0.36% error using two convolution layers in CNN. Our superior result is due to processing this dataset with balanced filters.

These analyses conclude that our FNN method helps design simpler CNN architectures for different classification tasks while producing good results. This evaluation reinforces the relevance of selecting adequate mask sizes and convolution unit numbers per CNN layer when representing input data. Additionally, adding new convolution layers does not affect results as expected.

compares error rates between our method and others across the three datasets used, showing the best experimental results. Architectures with single, double, and triple convolution layers are denoted as 1C, 2C, and 3C respectively.

---

4 Conclusion

This paper proposes a novel method to estimate convolution mask sizes and unit numbers needed to design simple CNNs for target classification tasks. Our approach, based on the false nearest neighbors (FNN) technique from dynamical systems theory, estimates sufficiently small embedding dimensions for task simplification, transforming relevant data into multidimensional space, and uses sparse filtering algorithms for unsupervised training. We disregard time dependencies, focusing instead on data (image) reconstruction. Our FNN method simulates reconstruction results of mask sizes in phase space. We also demonstrate that CNN convolution layers apply linear transformations to input data to produce a given shared domain, where vector neighborhoods are evaluated by our strategy. Our FNN method further estimates the number of convolution units comprising simple CNNs. Experiments confirm that our FNN method supports appropriate parameterization when designing simpler CNN architectures, finding good parameters for classification tasks while requiring fewer layers.

References

[1] Zhou Wenjie, Yan Jianfeng, Yang Lu. Research on prediction model of complaint based on deep learning [J]. Application Research of Computers, 2017, 34 (5): 1428-1432.

[2] Chen Yushi, Lin Zhouhan, Zhao Xing, et al. Deep learning-based classification of hyperspectral data [J]. IEEE Journal of Selected Topics in Applied Earth Observations & Remote Sensing, 2017, 7 (6): 2094-2107.

[3] Tian Zhuangzhuang, Zhan Ronghui, Hu Jiemin, et al. SAR ATR based on convolutional neural network [J]. Journal of Radars, 2016, 5 (3): 320-325.

[4] Gulcehre C, Firat O, Xu K, et al. On integrating a language model into neural machine translation [J]. Computer Speech & Language, 2017, 45 (C): 137-148.

[5] Li Xudong, Ye Mao, Li Tao. Review of object detection based on convolutional neural networks [J]. Application Research of Computers, 2017, 34 (10): 2881-2886.

[6] Lauer F, Suen C, Bloch G. A trainable feature extractor for handwritten digit recognition [J]. Pattern Recognition, 2007, 40 (6): 1816-1824.

[7] Lin Weichao, Ke S W, Tsai C F. CANN: an intrusion detection system based on combining cluster centers and nearest neighbors [J]. Knowledge-Based Systems, 2015, 78 (1): 13-21.

[8] Goodfellow I, Warde-Farley D, Mirza M, et al. Maxout networks [J/OL]. https://arxiv.org/pdf/1302.4389.pdf.

[9] Pierre Sermanet, Yann Lecun. Traffic sign recognition with multi-scale Convolutional Networks [C]// Proc of International Joint Conference on Neural Networks, 2011: 2809-2813.

[10] Yang M H, Dan R, Ahuja N. Learning to recognize 3D objects with SNoW [C]// Proc of European Conference on Computer Vision. 2000: 439-454.

[11] Albelwi S, Mahmood A. A framework for designing the architectures of deep convolutional neural networks [J]. Entropy, 2017, 19 (6), 239-242.

[12] Young S R, Rose D, Karnowski T, et al. Optimizing deep learning hyperparameters through an evolutionary algorithm [C]// Proc of Workshop on Machine Learning in High-Performance Computing Environments. 2015: 4-7.

[13] Rublee E, Rabaud V, Konolige K, et al. ORB: an efficient alternative to SIFT or SURF [C]// Proc of International Conference on Computer Vision. 2011: 2564-2571.

[14] Shotton J, Fitzgibbon A, Cook M, et al. Real-time human pose recognition in parts from single depth images [J]. Communications of the ACM, 2013, 56 (1): 116-124.

[15] Roweis S, Saul L. Nonlinear dimensionality reduction by locally linear embedding [J]. Science, 2000, 290 (5500): 2323-2326.

[16] LeCun Y, Bottou L, Haffner P. Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE, 1998, 86 (11): 2278-2324.

[17] Youness C, Asnaoui K, Ouanan M, et al. CBIR using the 2-D ESPRIT method: application to Coil_{100} database [C]// Computer Systems and Applications, 2016: 1-8.

[18] Hassairi S, Ejbali R, Zaied M. Supervised image classification using deep convolutional wavelets network [C]// Proc of International Conference on Tools with Artificial Intelligence, 2016: 265-271.

[19] Zhu Chenchen, Zheng Yutong, Khoa L, et al. Weakly supervised facial analysis with dense hyper-column features [C]// Proc of Computer Vision and Pattern Recognition Workshops, 2016: 93-101.

Note: Figure translations are in progress. See original paper for figures.