Hybrid-model for Handwritten Chinese Character Recognition based on Convolutional Neural Networks

Sheng-syun Shen[1], Hung-tsung Lu[2], Chih-hsiang Yang[1]

[1]Graduate Institute of Communication Engineering, National Taiwan University
[2]Graduate Institute of Computer Science and Information Engineering, National Taiwan University
R03942071, R03922011, R03942066

Abstract

Dealing with the Chinese handwriting recognition problem, we used support vector machine (SVM) and convolutional neural networks (CNN) to handle this task. There are many kinds of toolkits to implement these model recently, such as libSVM[1], libdnn[2] and Caffe[3], and we use all of them mentioned above on this final project work. However, because the training data may have much noise, we did some pre-processing like centralization, cropping, re-sizing and noise removal. To cope with the lack of training data, we tried to increase the data by adding virtual examples with rotation and masking. We also tried normalize the data non-linearly, and such a method performed well on the SVM structure. Experiments showed that the Caffe model architecture performed the best on most of the features we extracted. Finally, we tried the blending method on all CNNs trained above and got a surprising improvement of accuracy. We called such a CNN-blending model “hybrid-CNN model.”

Index Terms: libSVM[1], libdnn[2], Caffe[3], Support Vector Machine, Deep Neural Networks, Convolutional Neural Networks, Hybrid model

1. Introduction

The handwritten Chinese character recognition problem is one of the most popular research topics in image processing field. Several state-of-the-art approaches based on Convolutional Neural Networks (CNN) have been proposed in recent years. In this work, we also tried to start from the CNN architecture and implement some methods mentioned in our Machine Learning Techniques course, for instance, the support vector machine, the blending techinique and training with virtual examples. We got the best recognition accuracy after combining the blending method and virtual examples training into our Caffe CNN framework and we called this approach ”hybrid-CNN model.” The following sections will describe our implementation in detail. Section 2 is about the data processing which include some pre-processing methods and the virtual examples approaches. Section 3 is the description of our models. Section 4 shows the experiment results. Section 5 is our conclusions in this work.

2. Data Processing

This section will introduce all the data processing methods we have implemented, including pre-processing, feature transform, feature extraction, and so on.

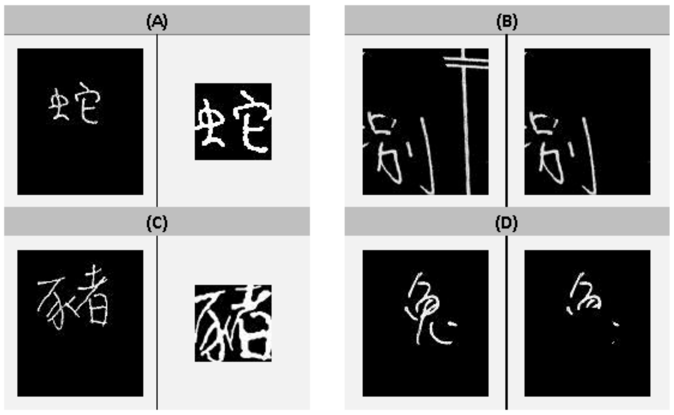

Figure 1: *This figure contains the original instances(left) and the data processing instances(right). (a)Centralization, cropping and re-sizing (b)Noise removal (c)Non-linear normalization (d)masking*

2.1. Data description

This data set is acquired from Machine Learning class which contains 22116 words, and 32 different type of handwritten Chinese character. Each instance has 12810 features with a range of [0,1], which is flatten by 105x122 height images. Most of the characters are incomplete cause they are partially covered by round masks. Due to the data noisy problem, we need to handle them before using the recognition models.

2.2. Centralization, cropping and re-sizing

At first, we discovered that after visualization, most of the Chinese characters are not at the center of the images. Moreover, some characters only hold a small portion in the 105x122 height images but some do not, which means the size of characters are different.

As a result, we proposed a method using MATLAB to adjust these data. We detected the location of every character, and moved it to the center of the image. Next, we cropped and resized the character to fit a 64x64 window to ensure every image be the same size. This method not only normalized the Chinese character instance, but also reduced the dimensions of feature vector and consequently reduced model complexity. The result is shown in Figure 1-(a).

2.3. Noise removal

While taking a glance at the training data, we found several cases of data noisy problem. One is the missing problem. Some instances lost their feature vectors and they should be removed from training data due to no training information. The other case, which we called ”double-line” problem, can be found if some double-lines across an training image. We were not sure what they are, but they didn’t belong to any kind of words, obviously.

To remove these double-lines, we therefore use a plug-in ”DBScan” in MATLAB, which is a density-based clustering algorithm, so that we can acquire strokes from every Chinese character image. If a stroke is longer than a threshold, it will be detected as a double-line because any Chinese character containing an extremely long stroke is an unusual phenomenon. The result after noise removal is shown in Figure 1-(b).

2.4. Virtual example: Rotation

Because of the lack of training data, we intuitively thought out a method to increase the training data – rotate the characters. When a person is writing, he/she may deviate the right angle, so the character will be slant. Based on this concept, we have tried many angles to rotate the training data: -3 to +3 degrees, and we found that rotate -2 to +2 degrees of training data cause the model overfitting. Therefore, we picked the rotating angle -3 and +3 to be the rotated virtual examples.

2.5. Virtual example and regularization: Masking

Due to the same reason in the previous sub-section, we need to increase the amount of training data to acquire higher training performance. Besides, we also faced some problem during the training process. The convolutional neural network models we used were very easily to over-fit the training data.

To solve these two problems, we came out with a novel method, which could support virtual examples and regularization techniques simultaneously. We know that most of the training instances were partially covered with small round masks at a random location. Therefore we place one more round mask on each character to add some noises, and then put these new instances back to the original training data. The radius of round mask we generate is 22 pixel, which is almost the same size as original instances, and the location of the centre is random. After the masking procedure, the size of training data is three times as the original one. The masking result is shown in Figure 1-(d).

2.6. Special: Non-linear normalization

Handwriting Chinese characters have complex structures and large shape variations[4]. Also, many similar patterns exist. Matching methods based on pixel values cannot perform well. Nonlinear normalization[5] is a useful technique for correcting nonlinear shape variations and homogenizing the two dimensional line density so that space can be more efficiently utilized and feature sampling points are stabilized for pattern matching methods. We implemented the nonlinear normalization in [4]. The output result is shown in Figure 1-(c).

3. Models

3.1. Support vector machines (SVM)

3.1.1. Model introduction

Support vector machines (SVMs) are well-known supervised learning models in machine learning. They are used for classification and regression analysis. Given a set of training data, a support vector machine tends to find a best hyper-plane to separate the data set and then classifies them. Besides linear classification, the support vector machine can efficiently perform a non-linear classification by using what is so-called kernel function, which mapped the data into a new feature space. The next subsection will show how we use the toolkit, libsvm.

3.1.2. Training architecture and implementation

In libSVM[1], we can choose many types of SVMs and kernels. The task is multi-class classification, so we choose C-SVC as our SVM type. The Chinese characters recognition is obviously not a linear separable case, and the characters are complicated, so we choose radial basis function (RBF) as the kernel type. The parameter cost and gamma on normalized training data after tuning is 4 and 0.00128 respectively; we try some parameter on all the other training data set include cost=4 and gamma=0.00128 and find out that these parameters are magic numbers which make the SVMs perform relatively well.

Besides the data processing we used as described in section 2, here we implemented the feature extraction in [4]. They divide a character image into 81 subarea, and each subarea is split into 4 parts, and accumulate the strengths of gradients with each 32 quantized gradient directions in each part. Then, downsampling the 32 dimensions vector to 16 dimension. Finally, we can get a 1296 (81*16) dimensions feature for one character. The result will show in next section.

3.2. Convolutional neural networks (CNN)

3.2.1. Model introduction

Convolutional neural network (CNN) is an advanced type of feed-forward artificial neural work, which is a biologically-inspired process and is a variation of multilayer perceptrons (MLP). In CNN models, the individual neurons are tiled in such a way that they respond to overlapping regions in the visual field. They are widely used models for image and video recognition, being a powerful tool for different vision problems.

3.2.2. Training architecture

The convolutional neural networks architecture is shown in Figure 2. There are some important structures of CNN we need to know:

- Convolutional layer: In a convolutional neural network, the parameters of each convolution kernel are trained by the backpropagation algorithm. There are many convolution kernels in each layer, and each kernel is replicated over the entire image with the same parameters. The function of the convolution operators is to extract different features of the input.
- Max Pooling: Another important technique of CNN is max-pooling, which is a form of non-linear downsampling. Max-pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value.

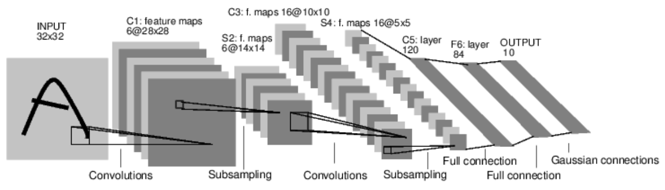

Figure 2: *Convolutional neural network model architecture.*

- Dropout: Because the fully connected layers almost copy all the parameters, it's really to over-fit the training data. Using the dropout technique, we can not only prevent over-fitting, but also speed up the training procedure. Dropout will perform randomly for every neuron. During the training procedure, if the neurons are dropped out, they won't be considered in forward pass and back propagation.

3.2.3. Implementation details

For every CNN models, we set the structure as 20x5x5-2s-20x4x4-2s-20x4x4-1023, three convolution layer and one fully connected layer. The down-sampling rate is 2, and learning rate is near by 0.005. To prevent over-fitting, we also set dropout layers between each convolution layer and fully connected layer. The dropout ratios are 0.1,0.1,0.1, and 0.25 respectively.

3.3. Caffe: A Deep CNN model with more Regularizatons

3.3.1. Model introduction and architecture

Caffe [3] is an open source toolkit for researchers to build DNN/CNN architectures intuitively and efficiently. In this work, we implemented a deep convolutional neural network model that A. Krizhevsky et al.[7] proposed which consist of 5 convolutional layers and 3 fully-connected layers with the dropout technique to prevent from overfitting. [8]

Before the start of training, the model rescales all training images into 256x256 size and crops each image into 5 parts: upper-left, upper-right, lower-left, lower-right, center, with the cropping size 224. In each feed-forward stage, the model randomly selects one of them and feed it into the neural networks to calculate and accumulate loss.

Here, in the feed-forward stage, instead of taking the widely used logistic sigmoids, we take rectifiers as our activation functions which can accelerate the model training speed without degrading the performance. [9] A rectifier is a activation function defined as

$$f(x) = max(0, x) \tag{1}$$

where x is the input to a neuron. A unit employing the rectifier is called a Rectified Linear Unit (ReLU). If at least some training examples produce a positive input to a ReLU, learning will happen in that neuron.

In order to aid generalization, local response normalization technique is implemented after applying the ReLU nonlinearity in first 2 convolutional layers. The response-normalized activity $b^i_{x,y}$ is given by the expression

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta} \tag{2}$$

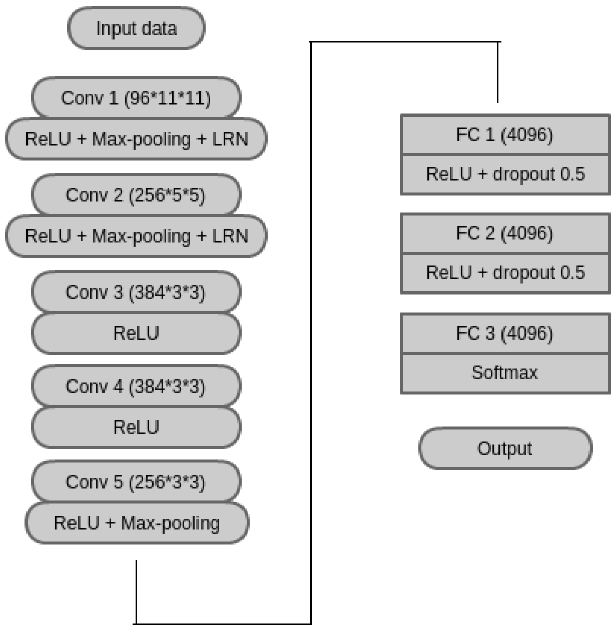

Figure 3: *Caffe architecture. N*M*M means that there are N kernels and the kernel size is M by M. LRN is the abbreviation of local response normalization and FC stands for fully-connected layer.*

where $a^i_{x,y}$ denotes the activity of a neuron computed by applying kernel i at position (x, y) and then applying the ReLU nonlinearity. The sum runs over n "adjacent kernels" and N is the total number of kernels in the layer. The constants k, n, α, β are hyperparameters. We set their values same as [7].

The idea of the local response normalization is inspired by the real neurons. It is a form of lateral inhibition, creating competition for big activities amongst neuron outputs computed using different kernels.

Except the settings described above, the remaining architecture of this Caffe model is similar to the CNN model mentioned in last subsection. The max-pooling method is also applied in some convolutional layers and each fully-connected layer has 0.5 dropout-ratio to prevent overfitting. The whole 8-layer deep convolutional model architecture is showed in Figure 3.

3.3.2. Hyperparameters tuning

Since the solver methods of deep convolutional neural networks address the general optimization problem of loss minimization which we often use SGD to handle, there are several hyperparameters that we can tune in this work. Stochastic gradient descent (SGD) updates the weights W by a linear combination of the negative gradient $\nabla L(W)$.

$$L(W) \approx \frac{1}{N} \sum_{i}^{N} f_W\left(X^{(i)}\right) + \lambda r(W) \tag{3}$$

$$V_{t+1} = \mu V_t - \alpha \nabla L(W_t) \tag{4}$$

$$W_{t+1} = W_t + V_{t+1} \tag{5}$$

where $f_W(X^{(i)})$ is the loss on data $X^{(i)}$ and $r(W)$ is a regularization term with weight λ. N is the mini-batch size in equation (3). V_t and W_t are the weight update and current weight at iteration t respectively. α is the SGD learning rate and μ is the

Group1: 一、二、三、六、七、八、九、十、牛、羊

Group2: 虎、兔、龍、馬、猴、雞、狗、豬、捌、參、陸、拾

Group3: 四、五、壹、貳、肆、伍、柒、玖、鼠、蛇

Figure 4: *New group for hybrid-CNN training.*

SGD momentum. We can tune different α and μ to get better model performance. Following are the 2 hyperparameters we have tuned.

- Learning rate: Conventionally, we usually set relative higher learning rate at early training and decrease it while training goes by. Choosing an appropriate learning rate is quite important. If we use a very large learning rate, at the SGD stage it will go a “big” step to update the model while if we use a very small value, the updating step will be too small. Both will make our model hard to find a local minima. After several experiments, we decide to set the base learning rate 0.001 and divide it by 10 every 50 epoches.

- Momentum: Momentum is a technique for accelerating gradient descent that accumulates a velocity vector in directions of persistent reduction in the objective across iterations. According to [10], we usually set the momentum value near by 0.9, which really works significantly in our project.

3.4. Hybrid-CNN model

3.4.1. Motivation and training architecture

While examining the mistakes on validation set, we discovered that there are ambiguous relationship between some kinds of characters. For example, Chinese character ”seven” is similar to ”ten” and ”one”.

To deal with this ambiguous relationship, we thus performed a two-layer hybrid-model . We first split the 32 kinds of Chinese character into 3 groups, which is shown at Figure 4, the training data are split into three groups as well. Then we train for each group using original CNN model to form the first layer. The second layer are trained by the whole training data. The testing detail of hybrid-model will be mentioned about in the experiment section.

4. Experiments

4.1. Feature Selection

In our first experiment, we want to see how different input features affect the model performance. So that we choose 4 kinds of input data with different pre-processing methods, as described following.

- Raw feature (Raw): Original data without any preprocessing

- Normalized feature (Norm): Original data with centralizing, cropping and resizing (described in section 2.2)

- Raw feature and remove noise (Raw+Rm): Original data with noise removal (described in section 2.3)

- Normalized feature and remove noise (Norm+Rm): Original data with centralizing, cropping, resizing and noise removal (described in section 2.2 and 2.3)

The results are showed in Table1 and Table2.

Table 1: *Feature Selection: Public Score*

Model/data type	Raw	Norm	Raw+Rm	Norm+Rm
SVM	0.651499	0.446734	0.653640	0.453694
CNN	0.206103	0.211991	0.204229	0.176660
Caffe	0.104925	0.116435	0.116167	0.111884

Table 2: *Feature Selection: Private Score*

Model/data type	Raw	Norm	Raw+Rm	Norm+Rm
SVM	0.662741	0.488490	0.653640	0.498662
CNN	0.197270	0.228854	0.217612	0.196467
Caffe	0.099036	0.126874	0.085385	0.117773

We can notice that no matter what input feature used, Caffe always performs the best and SVM is always the worst. It is not suprised that both CNN-based models apparently outperform the traditional SVM model.

In further comparison, the reasons that the Caffe model outperforms the traditional CNN can be contributed to some generalization techniques, e.g. 5 random cropping and the local response normalization, that Caffe takes.

We also discover that the Caffe model prefers raw data rather than normalized data while traditional CNN model prefers normalized data rather than raw data. One of the possible explanations is that Caffe model already has its own cropping strategies which the traditional CNN does not have and we are not necessary to centralize and crop images for Caffe model again.

Here, we will choose the best result for each model (Norm for SVM, Norm+Rm for CNN and Raw+Rm for Caffe) and discuss some regularization issues in the next experiment subsection.

4.2. Regularize / Virtual Example / Feature Transform

In this section, we compare the result between different virtual example and regularization techniques which is mentioned in section 2.4 to 2.6 and also check the affection after feature transform by implementing the method from Yamada et al. [5].

We can easily discover that the both virtual example and regularization techniques really help the improvement on recognition accuracy. Especially, Caffe can achieve an error rate lower than 10%, which is inspiring to us. The result shows that increasing the amount of training data can help machine learn better, and regularization will prevent CNN models from over-fitting during the training process.

While examining the Non-linear transform column, it’s surprised to see that SVM had a huge progress comparing to the original data set. Besides, Non-linear normalization on original CNN and Caffe model didn’t perform well. The SVM even beat the CNN model in this case. It’s interesting and we might discuss about it in the future.

Table 3: *Regularize/Virtual Example/Feature Transform: Public Score*

Model/data type	Rotation	Mask	Nonlinear
SVM	0.412473	0.422645	0.309690
CNN	0.153908	0.134368	0.447270
Caffe	0.097966	0.107602	0.148287

Table 4: *Regularize/Virtual Example/Feature Transform: Private Score*

Model/data type	Rotation	Mask	Nonlinear
SVM	0.448073	0.475642	0.338062
CNN	0.169433	0.150161	0.462527
Caffe	0.079497	0.087527	0.173983

4.3. Gradient feature extraction

In [4], a new feature is extracted. Table5 shows the results of the best parameters we found on the new feature extracted from Norm+Rm image and Nonlinear transform image.

Table 5: *Gradient Feature*

type/parameters	Public	Private
Nonlinear c=100 g=0.1	0.337259	0.3841012
Norm+Rm c=1000 g=0.1	0.268201	0.307013

Other results of different parameters didn't show on table. We have tried the cost form 0.01 to 1000 and gamma from 0.001 to 100 in log scale. The cost 10, 100 and 1000 coupled with gamma = 0.1 and 0.01 got the most comparable results. According to Table5, the gradient feature extracted from Norm+Rm is better than that from Nonlinear image, which is different from [4]. We thought that it is resulted from the variation of the training data. Our data has much noise and the characters are not unbroken. With the nonlinear transform, the characters became distorted and strange, which is the same reason that we thought why CNN not perform well on nonlinear transformed image.

4.4. Hybrid-CNN model and blending

In this section we preformed hybrid-CNN model using both original CNN and Caffe. The result is shown in Table 6.

For the three group models in the first layer, we selected the Norm+Rm with Mask for CNN as the training set. The training data will be split into three part according to the members of each group. We chose the same data set on the second layer CNN model, and this time we used the whole training data instead. After generating a hybrid-CNN model, we then performed blending. The other model we chose to blend is the caffe recognition result on Raw+Rm with Rotation data set. We blend the two model by weighted sum the two output vector. The weight for hybrid model is 0.25, and 1.0 for caffe model.

The public and private error rate outperformed all the other method we previously implemented. The tiny difference between public and private score also shows no over-fitting problem here.

There is still another way for error rate evaluation, which is called ”track 1” and we didn't mention on it. We believe that trying to optimize the error rate on ”track 0” only, the ”track 1” evaluation score will also be lower naturally. The error rate using ”track 1” measurement on our hybrid-CNN model is 0.213597 / 0.169433 (public / private).

Table 6: *Hybrid-CNN using blending result*

model	Public	Private
Hybrid-CNN with blending	0.078426	0.076017

5. Conclusions

In this report, we implemented many methods to recognize Chinese handwriting characters.

In data processing, besides raw data, we not only centralized, cropped and re-sized them but removed noise on both data sets. Moreover, to increase the amount of training data, we added virtual examples by rotating and masking. In addition, we referenced much papers to tune our hyperparameters on CNN and Caffe.

Furthermore, in order to improve the bad performance of SVM, we duplicated the work in [4], which used nonlinear normalization on image and extracted a gradient feature.

Experiments showed that CNN performed the best especially using Caffe toolkit, but it seemed that nonlinear transform on image is not suitable for CNN models. Increasing the amount of training data by adding virtual examples is work on all models. The gradient feature extracted form Norm+Rm image got the best performance on SVM although it is still worse than CNN.

Finally, most important of all, we blended many CNNs to built a hybrid-CNN model, and it got the most significant performance on both public and private scoreboards compared with all the other methods.

After several experiments, we can conclude that the CNN model is quite suitable for handling image processing tasks including the Chinese handwriting recognition task in this project. At the same time, the blending technique also demonstrates its power. We combine them together, and get a powerful hybrid-CNN model, which is quite an appropriate solver in this project. We recommend the hybrid-CNN model for both 2 tracks because of its powerful recognition rate and the relatively efficient training time.

6. Collaboration

All contents were discussed together. The following subsctions are the division of works of all implementations.

6.1. Sheng-syun Shen

- Data preparation
- Centralization, cropping and re-sizing
- Noise removal
- Virtual example and regularization: Masking
- Convolutional neural network model
- Hybrid model: Two-layer CNN model with blending

6.2. Hung-tsung Lu

- Deep convolutional neural network model (Caffe)
- Hyperparameters tuning

6.3. Chih-hsiang Yang

- Virtual example: rotation
- Nonlinear normalization
- Implement the work in [4]
- SVM baseline

7. References

[1] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

[2] Boton Chou, libdnn : a lightweight, user-friendly, and readable C++ library for deep learning, 2013-2014. libdnn on github: https://github.com/botonchou/libdnn

[3] Jia, Yangqing. "Caffe: An open source convolutional architecture for fast feature embedding." http://caffe. berkeleyvision. org (2013). Caffe on github: https://github.com/BVLC/caffe

[4] Dong, Jian-xiong, Adam Krzyak, and Ching Y. Suen. "An improved handwritten Chinese character recognition system using support vector machine." Pattern Recognition Letters 26.12 (2005): 1849-1856.

[5] Yamada, Hiromitsu, Kazuhiko Yamamoto, and Taiichi Saito. "A nonlinear normalization method for handprinted Kanji character recognitionline density equalization." Pattern Recognition 23.9 (1990): 1023-1029.

[6] "Convolutional Neural Networks (LeNet) DeepLearning 0.1 documentation", in *Deep Learning*, Retrieved January 20, 2015, from http://deeplearning.net/tutorial/lenet.html

[7] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[8] Hinton, Geoffrey E., et al. "Improving neural networks by preventing co-adaptation of feature detectors." arXiv preprint arXiv:1207.0580 (2012).

[9] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010.

[10] Sutskever, Ilya, et al. "On the importance of initialization and momentum in deep learning." Proceedings of the 30th International Conference on Machine Learning (ICML-13). 2013.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database. IEEE Computer Vision and Pattern Recognition (CVPR), 2009. http://www.image-net.org/