Information Based Universal Feature Extraction

Mohammad Amiri and Rüdiger Brause
Goethe University, Computer Science Dep., Frankfurt, Germany

ABSTRACT

In many real world image based pattern recognition tasks, the extraction and usage of task-relevant features are the most crucial part of the diagnosis. In the standard approach, they mostly remain task-specific, although humans who perform such a task always use the same image features, trained in early childhood. It seems that universal feature sets exist, but they are not yet systematically found. In our contribution, we tried to find those universal image feature sets that are valuable for most image related tasks. In our approach, we trained a neural network by natural and non-natural images of objects and background, using a Shannon information-based algorithm and learning constraints. The goal was to extract those features that give the most valuable information for classification of visual objects hand-written digits. This will give a good start and performance increase for all other image learning tasks, implementing a transfer learning approach. As result, in our case we found that we could indeed extract features which are valid in all three kinds of tasks.

Keywords: Machine Vision, Universal feature extraction, Information Theory, Transfer learning.

1. INTRODUCTION

Humans have sought to extract information from imagery ever since the first photographic images were acquired [1]. The most useful basic components are called *features*. Feature extraction and representation are crucial steps for object recognition. One issue is the effective identification of important features in images, and the other one is extracting them. It is a difficult task to obtain a prior knowledge of what kind of information is required from the image, even when you know the image domain. Much of the information in the data set may be of little value for discrimination. However, there is a general agreement that the tools available for analysis of images are not sufficient. It is still a challenging problem in computer vision how to extract *universal features* that reflect the fundamental substance of images as complete as possible.

In this context, we might ask: Do *universal features* in images exist such that by using them we are able to efficiently recognize any unknown object? Is it necessary to extract new special features for any new object recognition tasks? Are there some general features in natural and non-natural images which can also be used for specific object recognition? Very little research attention has been paid to these problems in the last decades. Some people used the concept of *transfer learning* to reuse the knowledge which taken from one classification problem for similar problems [2]. Raina et al. [3] also used a similar paradigm which is *self-taught learning*, to use the knowledge of some non-labeled data for supervised classification of groups of animals with limited number of images.

2. UNIVERSAL FEATURE EXTRACTION

In this section, we will develop the notation of universal features. How can we show that a feature is universal or not? One criterion is its applicability: a universal feature has to be effective in all applications ever existed and yet to come up. Unfortunately, there is no practical way to prove this. Instead, we will construct it by theoretical considerations and then show the feature's e effectiveness. Alternatively, we may not need to prove that a certain feature is universal; it rather means that it is not specific to any particular application. In this contribution, we will focus on the question: What kind of features are the best for classifying objects? It is well known that the best strategy for classification is the Bayes decision criterion [4]: given an image x, choose that class W_k which has the highest conditional probability of occurrence. Unfortunately, we do not know the conditional probabilities. Instead, we have to estimate them.

Let us assume that we observe pictures x containing an object. Additionally, a teacher will tell us with the decision L if the object is present: $L = 1$ indicates yes and $L = 0$ means *no*. Therefore, the observation set consists of pairs $(x; L)$. Now, instead of using the whole picture only a small set of features $h_1,\ldots,h_n$ extracted from x by a function $h(x)$ should be sufficient for detecting the object. How can we find it? Let us first consider just one feature *h*. This means, that the probability of the correct decision for the presence of object *P (L|x)* should be as close to *P (L|h)* as possible. Since the probability for correct classification is based on the conditional probabilities, the distance between the two probability distributions can be seen as a measure for the classification quality. It is well known that the Kullback-Leibler distance

$$D(P(L|\mathbf{x}); P(L|y)) = \sum_L P(L|\boldsymbol{x}) log \frac{P(L|\boldsymbol{x})}{P(L|y)} \qquad (1)$$

becomes only zero if and only if the two probability distributions become equal. Now, we have a problem: since *h(x)* is an unknown function, we do not know *P (L|h)*. Instead, we can estimate it by a function *g(L|h)* which does depend on the decision *L*, but is indeed a function of *h* only. Nevertheless, if we maintain $0 < g < 1$ the Kullback-Leibler distance will still become only zero if the two probability distributions become equal. Therefore, we might use the expected distance as an objective function. Instead of minimizing the whole expression, we take the denominator as objective function *R* to be maximized for setting up the unknown function. It can be shown that *R* becomes for *M* samples

$$R\,(g,h) = \frac{1}{M} \sum_{i=1}^{M} L(i) log\ g(h)\ +\ (1 - L(i) log\big(1 - g(h)\big) \qquad (2)$$

This function is well known as maximum likelihood objective function. Now, how can we obtain the unknown functions *g* and *h*? Let us assume that we use parameterized functions, i.e., the necessary information for extracting and using the features are stored in a finite set of parameters.

The object detection function *g(y)* is determined by ***s*** parameters $g(\mathbf{h}(\boldsymbol{u}),\boldsymbol{w})$ with $\boldsymbol{h}=(h_1,..,h_m)$ and $\boldsymbol{w}=(w_1,\dots,w_s)$.

Thus, the task of determining the universal features becomes a task of determining the appropriate parameters of the unknown functions.

3. LEARNING THE FEATURE EXTRACTION

In this section, we will describe our approach for extracting the universal features by minimizing the objective function. Unfortunately, the desired solution is problem dependent, i.e. it depends on our observation set. One common approach for minimizing an objective function, if there is no analytic solution, is the stepwise iteration of an approximation expression, a so-called learning algorithm, using the observations as training set.

As learning algorithm for the first and second sets of parameters ***w*** and ***u*** we might use the well-known stochastic gradient ascend for maximizing *R*,

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \gamma(t) \frac{\partial R(g(\boldsymbol{w}))}{\partial \boldsymbol{w}} \quad \text{and} \quad \boldsymbol{u}(t+1) = \boldsymbol{u}(t) + \gamma(t) \frac{\partial R\big(g(\boldsymbol{u})\big)}{\partial \boldsymbol{u}} \qquad (3)$$

3.1 The Neural Net for Extracting One Feature

Now, we have to choose the kind of functions *g(.)* and *h(.)* to use here. It is well known that all continuous functions can be approximated sufficiently well by two layer networks using sigma neurons and squashing functions *S* as output functions [5]. Therefore, choosing our approximation functions like this will not limit our approach in any way. With the image input described by a pixel tuple **x**, we might choose as extraction function a squashing function with an affine argument $h_j(\boldsymbol{u},x)=S\,(\boldsymbol{u}^T\mathbf{x})$ and as object detection function for one object $g(\mathbf{w};\,\mathbf{h}) = S_F\,(v)$ with $S_F(v) = \frac{1}{1+e^{-v}}$ and $v=\boldsymbol{w}^T h$.

This can be interpreted like that we have a first layer of formal neurons, implementing sigma neurons and squashing function $h(\mathbf{u},\mathbf{x})$, and a second layer, implementing the object detection function *g(h, w)*. In Figure 1 the two layer architecture is shown with several output units, each one detecting a different object. Now, to obtain the desired iteration equations, the learning rules, we use the standard back-propagation approach for our risk function and compute the necessary derivatives.

For learning, we assume several important properties:

- Each extraction function h_j covers a different part of input **x**, i.e., it has an unique receptive field (RF) and is not completely overlapping with other fields, see Figure 1. This means, that the tuple of input pixels **x** is different for each extraction unit *j*, denoted by $\mathbf{x}_j$.

- The object should be recognized everywhere on the image. Therefore, in order to train only the statistics and avoid over- fitting, we put the constraint that the parameters $\boldsymbol{u}$ of each extraction function are the same ones, i.e., all hidden neurons share the same k weights.
- There can be more than one object present, i.e. N ones which should be recognized independently. Therefore, we assume not one, but N functions g_k, i.e., N output units.

An important decision in this network is that we use the weight sharing idea in the feature extraction layer. Using weight sharing has two advantages: First, it reduces the number of parameters for learning, and second, all neurons detect the same features, although their receptive fields are located at different positions in the input image.

The weights **w** in the last layer are not shared. The number of output neurons depends on the number of classes (sets) that we need or how many sets we want for classification. In figure 1(left) the overall architecture is shown.

We use the first layer ($\boldsymbol{u}$) as feature extractor and the second layer ($\boldsymbol{w}$) as classifier layer. Since all outputs $g_k(\boldsymbol{w}_k; \boldsymbol{h})$ can be computed independently from each other, the stochastic gradient learning rule does not change much.

The input samples are no longer treated similarly by the extraction units $h_j\ (\boldsymbol{x})$, but they are grouped into subsets. Each unit j processes only a subset $\boldsymbol{xj}$. The input samples can be arranged in different manners. In Figure 1(right) the samples are arranged in a two-dimensional manner, e.g. like pixels of an image. As you can also see in Figure 1 (right), we extract several patches from each image and use them as inputs for the network. The number of patches that can be extracted from an image depends on some factors, e.g. the size of a patch, the size of the image and the number of pixels shared between two neighbor patches. For instance, the number of rectangular patches which can be taken from an image with 60*80 pixels and a patch size of 9*9, sharing three pixels, are 108. Please note that we extract square patches instead of circular ones because it is computationally more feasible. There are still some open questions for this architecture: What is the best size of a receptive field and what is the optimum number of hidden units? We will discuss these questions in later sections and present some experimental results. It is clear, that by increasing the size of the image, we need more receptive fields and more parameters in the subsequent layer. Instead, it might be better to increase the receptive field size for covering the image by a smaller number of fields. Additionally, by having more layers we need more training example to learn more parameters. If we do not have them, we have to reduce the number of parameters

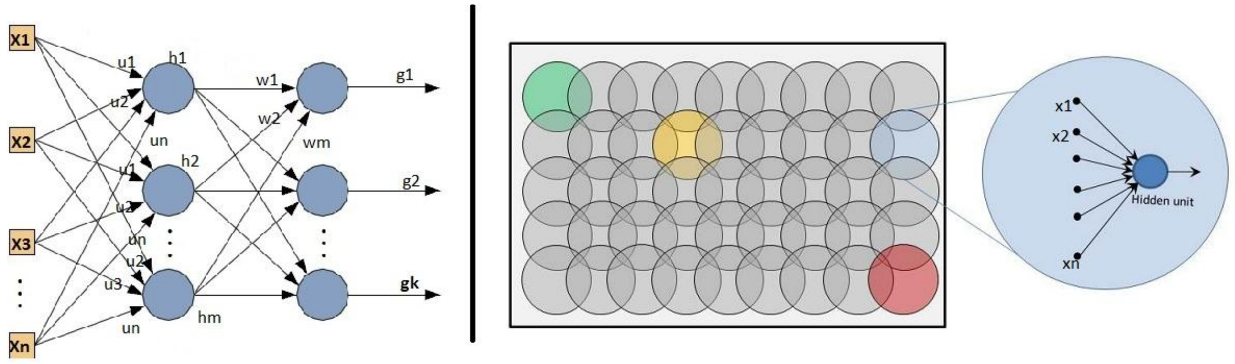

Figure 1. left): The network architecture for function approximation (from [6]) right): The RF patch extraction from an image

3.2 Extracting Several Features

Our feature extraction analysis of the previous section only covers just one feature in each RF, the most important one. How do we get additional, helpful features? Let us assume that in each RF we extract not only one feature, but r ones. Then, each extraction result h_j of RF j has several components

$$h_j = (h_1(\boldsymbol{u}_1; \boldsymbol{x}_j), \dots, h_r(\boldsymbol{u}_r; \boldsymbol{x}_j))^T \text{ with } h_i(\boldsymbol{u}_i; \boldsymbol{x}_j) = \boldsymbol{S}(z_{ij}); z_{ij} = \boldsymbol{u}_j^T \boldsymbol{x}_j \tag{4}$$

The activity of the second layer, the object detection, will not change except of the fact that for each output unit the number of inputs becomes $m*r$ instead of m.

Certainly, the learning equations change with the additional features. The equation has now $m*r$ components, and eq. 3 becomes for the $s\text{-}th$ feature

$$\boldsymbol{u\_s}(t+1) = \boldsymbol{u\_s}(t) + \gamma(t)\frac{\partial R(g(\boldsymbol{u\_s}))}{\partial \boldsymbol{u\_s}} \tag{5}$$

For N outputs, eq. 3 changes to

$$\boldsymbol{w\_k}(t+1) = \boldsymbol{w\_k}(t) + \gamma(t)\frac{\partial R(g(\boldsymbol{w\_k}))}{\partial \boldsymbol{w\_k}} \qquad (6)$$

Now, there are *r* features learned by each of the receptive fields. How can we assume that they will be different although they have the same input and the same learning rules? The answer lies in the fact that all feature vectors us have different feedback from the second layer, depending on their own activity h_s. This leads to different learning behavior and different convergence states.

4. EXPERIMENTAL RESULTS

In this section we report about several results using the ideas and algorithms presented so far. First, we discuss the setup of the training procedure and some of the network parameters always used. Then, the results of training and testing with different kind of images and parameters are reported.

4.1 Network Parameters

To prepare the network for training, several decisions have to be taken before. First, let us discuss the choices which are constant during training and testing.

Activation Functions: There are number of common activation functions in use for artificial neural networks. In our case, we used the bipolar *tanh* activation function for the hidden layer units of the network and the unipolar sigmoid function for the output units of the second layer, because the output should show the amount of probability that an input object may be in a class. Therefore, it has to be between zero and one.

Weight Initialization: There are also many possible algorithms for initializing the weights for feed for-ward neural networks [7]. We used the method which there is just a maximum bound for the weights and the initialization is still random, because it is simple and our experimental result showed that it performs better than the other methods.

$$|w_{ij}| < 2.4/N\_{input} \qquad (7)$$

The variable N_{input} refers to the number of input units. This value is in our case the image patch size of 9* 9 + 1 = 82. The additional one is for bias.

Learning Rate: In all tests in training and test phase the learning rate is = 0.005.

4.2 Input Data Processing

Some object images are taken from the Amsterdam library of object images (ALOI) database. ALOI is a color image collection of 1000 small objects, recorded for scientific purposes. In order to capture the sensory variation in object appearance, they systematically varied viewing angle, illumination angle, and illumination color for each object, and additionally captured wide baseline stereo images. They recorded over a hundred images of each object, yielding a total of 110,250 images for the whole collection. Objects are artificial (e.g., a hat or a cup) and natural (e.g. an apple or an orange). We placed the selected objects in the middle of some natural and artificial background images and a variation of three pixels shifted left, right, up or down, maximally. By this, we prepared nine sets of data. For preprocessing the input images, we normalized each input pixel set x to zero mean and unit variance of all pixel values. The size of the objects to recognize is 80*60 pixels. These objects are placed before different backgrounds.

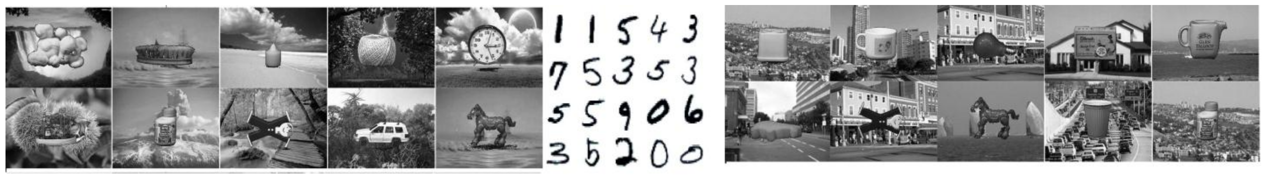

Figure 2. Examples objects located in natural background images (left), some examples of MNIST handwritten digits (middle) and objects located in artificial background images(right)

For example, Figure 2 shows different objects with multiple illumination and viewing angles, placed before different background images. Additionally, we used the MNIST database of handwritten digits which has a training set of 60,000 examples, and a test set of 10,000 examples. The digit images have 28*28 pixels [8]. Before use, we normalized the size and centered it in a fixed-size image. This set was chosen to represent unnatural objects. If features are universal, they have to represent efficiently also those objects.

The following scheme gives an overview of the composition of the different training and test sets.

TABLE 1: THE COMPOSITION OF THE TRAINING AND TEST SETS

Set label	Object	Background
Set1	Artificial	Natural
Set2	Artificial	Artificial
Set3	Artificial	Natural
Set4	Artificial	Artificial
Set5	Natural	Natural
Set6	Natural	Artificial

For training, in set 1 objects are shifted a little from the center and they have different view and illuminations with many natural backgrounds. In set 2, the same objects are used for set 1, are treated similar using non- natural backgrounds. Set 3 is made by treating other artificial objects in natural backgrounds and set 4 are made by putting these artificial objects in non- natural backgrounds.

For test, in set 5 and 6 objects are natural (like apples or potatoes) with the same backgrounds as in set 1 and set 2.

4.2 Does the Network Learn Universal Features?

We trained the system with 40,000 input images in ten different groups with 16 neurons in the hidden layer. Each group included one object with multiple illuminations and view angles, placed in the middle of many background images of set 1, with a maximum of three pixel shifts in right or left and up or down. The size of a RF was set to 9 by 9, and each RF shared three pixels with its neighbors. This value was determined experimentally; it gives better result than others. After convergence of the network, we fixed the value of the first layer (U), the features, and used it as feature extractor for further processing.

The weights of the second layer were trained separately to classify multiple objects with multiple backgrounds. After training, it could classify ten groups (according to the objects in the images) of data set images. For evaluation, we used as classification accuracy

$$\mathit{Accuracy} = 0.5*(\mathit{Prob}\ (PT) + \mathit{Prob}\ (NF)) \qquad (8)$$

Please note that, for calculating the rate of accuracy, we had to record the positive (true) PT and negative (false) NF system classification decisions. If we just use the positive input PT rate to compute the accuracy rate, by changing the threshold value we can get better result. Therefore, for a fair comparison, we had to take both rates into account. In general, a ROC analysis have to be computed, but the averaged correct decision is sufficient for this application. For more information about ROC analyses see [9]. For computing the probabilities, we used the classification output of the neural network units. Because the output of the units is between zero and one, to assign an object to a class we selected the maximum value of the output. Thus, the object is the member of a class with the maximum output value. The test revealed that with 99.2% accuracy set 3 was correctly classified, and with 94.66% accuracy set 5. Set 3 includes artificial objects and set 5 includes natural objects.

After this, we set up the second layer to classify the MNIST handwritten digits by using 60,000 data for training and 10,000 data for test. After this, it could classify ten groups (0-9) of the handwritten digit images of the test set with 92.83% accuracy. It is interesting to know that by using the MNIST exclusively for training the features, the rate of correct accuracy was 95.32%. The small difference between the results shows that both sets had the same statistical proportions, giving rise to quite optimal features used for digit classification. In comparison to this, the result of the handwritten digit recognition by the LDA classifier implemented in the Matlab software package was only 87.6%. The best result for handwritten digit classification reported in literature is 99.77% for the training error and was obtained using a special 6 layer non-linear neuron network, stacked on top of each other (convolutional neural network) [10]. Consider that this result was not obtained by universal features and their test set results should be worse, our results are very good. The state of the art result for ALOI dataset classification is 99.8% [11].

4.3 Effect of Change in the Size of Receptive Field

Changing the RF size to 19*19 and the RF share to 6 results in an accuracy of 95.95%, 94.33% and 90.20% respectively for set 3, 4 and 5. We also changed the RF size to 7*7 and RF share to 2. The result was 98.98%, 97.93% and 94.57% respectively for set 3, 4 and 5. It means that the best size for RF is 9*9.

4.4 Changing the Number of Features (Hidden Units)

In this test all configuration and initialization was done as in section 4.3 except that we changed the number of features from 16 to 9. The resulting accuracy was 98.75%, 98.67% and 94.5% respectively for sets 3, 4 and 5. By decreasing the hidden unit from 16 to 9, the rate of accuracy is also reduced a little bit. In all of above tests we used set 1 for training the features.

5. CONCLUSION

In this paper we proposed a new method for universal feature extraction. First, we used an information theory approach to design a proper risk function which leads to cross entropy minimization. We developed a feed forward neural network as basic structure to extract the universal features. Second, as constrain we used a weight sharing method for all receptive fields. In addition of reducing the number of learning parameters it has the benefit that the shared weights makes all neurons detecting the same features, independent of their different positions in the input image. The results show the success of this method in some applications e.g., hand written digit classification and recognition of natural or artificial objects which are placed in the natural or artificial background images.

There are also some questions which will be studied in future: 1) How can we make the system invariant to the position of objects in image so it could recognize objects not only in the center of background image, but also in any places of image? 2) How can we make the system to adapt to different shading and object size? How can the optimal size of the receptive fields be obtained automatically? Here, more dynamical architectural approaches have to be developed.

REFERENCES

[1] Lindi J. Quackenbush,"A review of techniques for extracting linear features from imagery," Photogrammetric Engineering & Remote Sensing, 70(12):1383-1392, December 2004

[2] Dan C. Cires and Ueli Meier, "Transfer learning for latin and chinese characters with deep neural networks," In in Proc. IJCNN, pages 1-6, 2012.

[3] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data," In Proceedings of the 24th international conference on Machine learning., pages 759-766. ACM, 2007.

[4] R. Duda, P. Hart, and D. Stork, "Pattern classification," Wiley & Sons, New York, 2000.

[5] K. Hornik, "Approximation capabilities of multilayer feedforward networks," Neural Networks, 4:251-257, 1991.

[6] M. Haibach "Informationsgesteuerte, adaptive extraktion von merkmalen" Master's thesis, Computer Science Dep.,Goethe-university,Germany, 2011.

[7] M. Fernandez-Redondo and C. Hernandez-Espinosa, "Weight initialization methods for multilayer feedfor-ward," ESANN'2001 proceedings, pages 119-124, April 2001.

[8] Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," In Advances in Neural Information Processing Systems, pages 396-404, 1990.

[9] Charles E. Metz, "Basic principles of roc analysis," Seminars in Nuclear Medicine, 8(4):283298, October 1978.

[10] Jurgen Schmidhuber, "Multi-column deep neural networks for image classi cation," In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3642-3649, 2012.

[11] Philippe Smagghe, Jean-Luc Buessler, and Jean-Philippe Urban, "Novelty detection in image recognition using IRF neural networks properties," In ESANN-2013, 2013.