Graph-Based Supervised Clustering in Vector Space

Lily Schleider[1,2(✉)], Eduardo L. Pasiliao[1,2], and Qipeng P. Zheng[1,2]

[1] University of Central Florida, Orlando, FL 32816, USA
lily.schleider@knights.ucf.edu
[2] Air Force Research Laboratory, Munitions Directorate, Eglin AFB, FL 32542, USA

Abstract. Neural Networks are able to cluster data sets and our goal was to figure out how well the neural network clustered. The MNIST data set was ran through a neural network and the distances were extracted from both the feature and output layer. Five different distances were used on both layers. K-means clustering was used assess the clustering performance of each layer in the neural network. Results conveyed that the feature layer was not as proficient at clustering when compared to the output layer. The type of distance did not make a significant difference for clustering. These conclusions can be derived from qualitative observation of the cluster graphs. By observing the clustering performance of the different layers in the CNN, we are able to gain insight on the neural network.

Keywords: Neural networks · Feature layer · Output layer · Clustering

1 Introduction

Machine learning allows computers to "learn" from a training data set so it can make decisions and/or predictions. Neural networks are a fundamental aspect of machine learning. They work by recognizing patterns. A convolutional network applies a filter over the images and create a feature map. Convolutional neural networks (CNN) can have multiple filters and multiple feature maps. A max-pooling layer is applied in between filter convolutions in order to keep the most significant features while reducing the dimensions. The feature map created by the convolutions allow the machine to learn traits in the data set. Once the convolutions and maxpooling is done, everything is flattened into one dimension that is fully connected, which is the output layer.

Mao and Jain [3] designed a Linear Discriminant Analysis (LDA) network, Sammon's projection network, a Kohonen-Based Nonlinear Projection Network (NP-SOM), and a Nonlinear Discriminant Analysis Network (NDA). All of these networks utilize adaptive learning and are ideal for data sets where patterns

S. Chellappan et al. (Eds.): CSoNet 2020, LNCS 12575, pp. 476–486, 2020.
https://doi.org/10.1007/978-3-030-66046-8_39

change over time. They found that the NP-SOM was the best at data visualization while the Sammon's network was the best at maintaining cluster shape, data structure, and inter-pattern distances. The NDA was the best at classifying and the results from this paper were consistent with other analysis.

Research has been done that compares the performance between auto-learned features from Deep Convolutional Neural Networks (DCNN) and hand-crafted features. Their goal was to find a noninvasive method for detecting Circulating Tumor Cells in a wide variety of cancer types. The DCNN is trained from positive, negative, and false positive data sets. The false positives are classified into either hard or easy using K-means clustering method. The hard samples are added to the negative data set. The hard-false positive samples are given more weight during training in order to allow the classification boundary to get closer to the hard samples. They use sigmoid activation function in their convolutional layers. The hand-crafted features are made by using the Histogram of Gradients and Histogram of Color in the Support Vector Machine. It took the DCNN longer to run but it also performed better than the hand-crafted features. The DCNN also reduced the redundancy in the samples [4].

Pradheep and Srinivasan's [5] paper introduces diagonal based feature extraction and compares it to horizontal and vertical feature extraction. They used images of off-line handwritten letters that were written by different people as their data set. These images were scanned and had to be pre-processed before feature extraction. The features are extracted by moving across the diagonals in each zone, which are size 10×10 pixels. Each zone gets 19 sub features and they are averaged to get a single value of the entire zone. Next, a feed forward back propagation neural network executes the classification. They use log sigmoid activation function. This paper concluded that the diagonal feature extraction had a 97.84% accuracy and worked better than vertical or horizontal method.

K-means clustering organizes observations into clusters based on their means. The k in k-means clusters is the number of clusters you are trying to organize the data into. The algorithm selects a random 10 data points at first (we can call these the initial clusters). It then measures the distance between every other point and the initial cluster and groups it with the nearest of the initial clusters. It then finds the total variance in each of the clusters. It repeats these steps until the variation in each cluster is close to each other. If we wanted to find the best k, then we would plot the reduction in variation using different values of k and see where there is significant reduction of variance and choose that k. This is called an elbow plot. In our case, the MNIST data set has its predetermined clusters so we know $k = 10$ [2].

The primary focus of this paper is to determine how well the both the feature layer and the output layer of the CNN is able to cluster using k-means clustering. This would allow us to see at which stage the clustering is most effective and can give us an insight as to why it does. We can see how significant different layers and different distances are for the CNN to cluster. This would give us more information on CNNs and how they work.

2 Experiment

2.1 Dataset

The specific data set we worked with came from the Modified National Institute of Standards and Technology (MNIST) database. It consists of 60,000 training images and 10,000 testing images. The images are black and white and have handwritten digits 0–9 with a white background. Each image is 28×28 pixels. We had to have the images in forms of arrays so we could find the distance between the images. The arrays had numbers ranging from 0–255 with smaller numbers representing lighter shades and larger numbers representing darker shades. For example, the white background of the images would have 0's. We divided the numbers by 255 so we could normalize the numbers and have them be between 0–1.

Our goal was the figure out which images have edges in between them. The edges represent similarities between the images. Once we find the similarities, we can cluster the images based on their digit (Fig. 1).

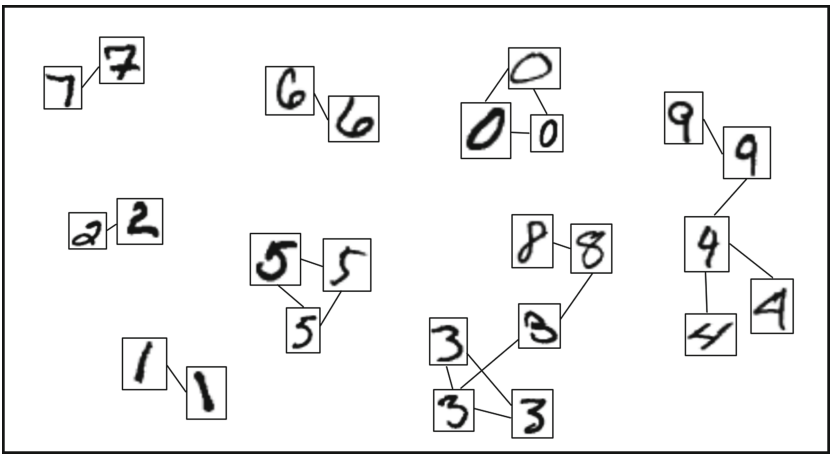

Fig. 1. This is a representative diagram. Notice how certain numbers have edges with other numbers that are a different digit (a 3 has an edge with an 8). This is an example of false alarms. There are also edges that should exist but do not (two 4's do not have an edge). This is an example of missed detection.

Our first step was the run the MNIST data through a CNN so we can extract the distances in both the feature layer and the output layer. The images were in the form of an array so we can find the distances easily. We used 5 different types of distances: dot product, Euclidean distance, Minkowski distance, Manhattan distance, and Chebyshev distance. We use these distances to create a k-means clustering graph. Jupyter Notebook was used for the clustering and Spyder was used for the CNN and distances. The Scikit-learn library was used for the clustering and Keras was used for the CNN (Fig. 2).

2.2 Distances

$$\boldsymbol{v}_i \qquad \leftarrow \text{Image vector } i \tag{1}$$

$$\mathrm{D}_p(\boldsymbol{v}_i, \boldsymbol{v}_j) = \left(\sum_k |\boldsymbol{v}_{ik} - \boldsymbol{v}_{jk}|^p \right)^{1/p} \quad \leftarrow \text{Minkowski distance} \tag{2}$$

$$\mathrm{D}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \sum_k |\boldsymbol{v}_{ik} - \boldsymbol{v}_{jk}| \qquad \forall i \leq j \leftarrow \text{Manhattan distance} \tag{3}$$

$$\mathrm{D}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \left(\sum_k |\boldsymbol{v}_{ik} - \boldsymbol{v}_{jk}|^2 \right)^{1/2} \qquad \forall i > j \leftarrow \text{Euclidean distance} \tag{4}$$

$$\mathrm{D}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \max_k |\boldsymbol{v}_{ik} - \boldsymbol{v}_{jk}| \qquad \forall i \leq j \leftarrow \text{Chebyshev distance} \tag{5}$$

$$\mathrm{D}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \sum_k (\boldsymbol{v}_{ik} \cdot \boldsymbol{v}_{jk}) \qquad \forall i \leq j \leftarrow \text{Dot product distance} \tag{6}$$

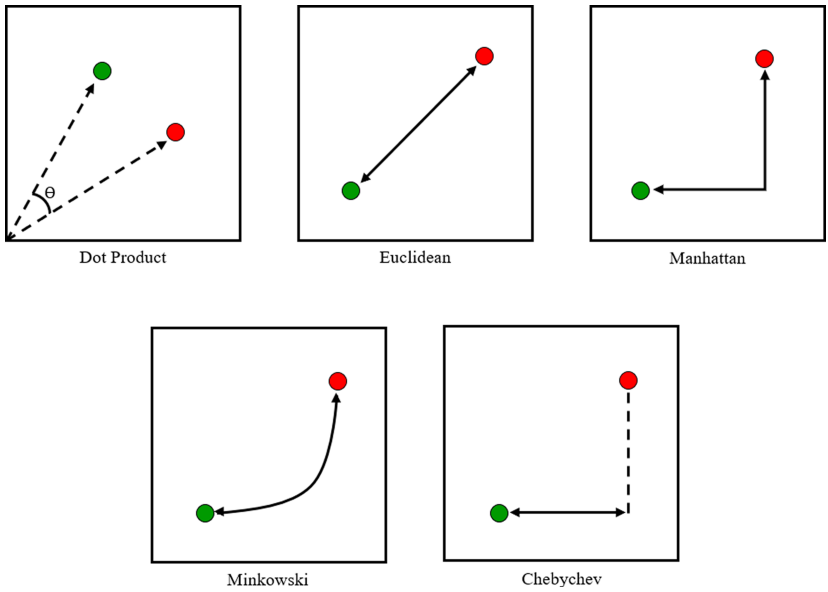

Fig. 2. Visual representation of the different type of distances that were used [1].

The dot product (6), also known as the scalar product, uses the length of two vectors and the angle between them. The Euclidean distance (4) finds the direct distance between two points in the vector space. The Manhattan distance (3) is similar except it only measures along vertical and horizontal axes with right angles. It is also known as the city block distance because it measures as if it is traveling across a city block grid. The Chebychev distance (5) calculates either the x distance or y distance depending on which is greater. It is also known as the chessboard distance. The Euclidean, Manhattan, and Chebychev distances are subsets of the Minkowski distance (when $p = 2$, $p = 1$ and $p = \infty$, respectively). The Minkowski distance (2) is determined in a normed vector space and can take in any p value. For our Minkowski distance, we used $p = 3$.

2.3 Convolutional Neural Network

Keras was used to create the CNN. The CNN was trained using the MNIST dataset. The CNN consists of a convolution layer, maxpooling layer, flatten layer, and two dense layers. The activation function relu was used in the convolutional and the first dense layer. Relu is a common activation function in neural networks and is defined as $y = max(0, x)$. The softmax activation was used in the last dense layer. Softmax is a normalized exponential function. It creates a normalized probability distribution. The feature layer and output layer were extracted and the distances were found for both of them. The flatten layer was retrieved from the flatten layer while the output layer was retrieved from the second dense layer.

3 Data and Results

The output layer distances were able to be clustered much better than the feature layer distances. We can use qualitative observations to see that the feature layer muddles the clustering while the output layer does a much better job at clustering. The clustering can be judged based on the clarity and distinctness of the clusters in the graph. A more distinct graph indicates better clustering.

The clustering from the feature layer displays errors and shows a lack of accuracy (Fig. 3, 4, 5, 6 and 7). There aren't any definite clusters. From the output layer clustering we can tell that the clusters are clear and defined (Fig. 7–8). Each digit has a clear cluster. The clustering is still not perfect. In the dot product distance metric, you can see that there is still some misclustering. For example, a zero is being misclustered as a 5 (Fig. 12). Overall the output layer was able to be clustered very well (Figs. 9, 10 and 11).

By qualitative observations of the graph, we can see that once we reach the output layer, the neural network was very good at clustering the MNIST data set, but it was not 100% accurate. As we saw, the output layer still made a couple of errors. The main difference between the layers is that there is a softmax function in the output layer, which was the key difference in improved clustering. There was only a significant difference in clustering when using dot product with the output layer. That was the only distance that made a couple of clustering errors. Other than that, the distance did not make a overwhelming difference.

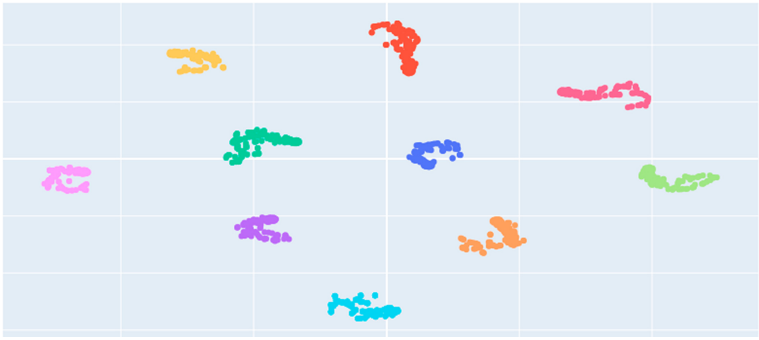

Fig. 3. Dot product on feature layer

Fig. 4. Euclidean distances on feature layer

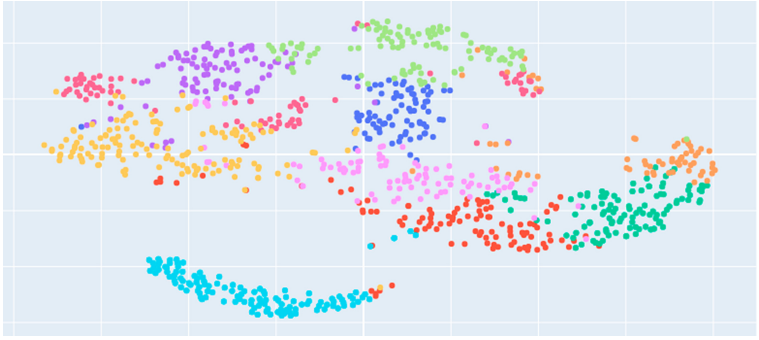

Fig. 5. Minkowski distances on feature layer

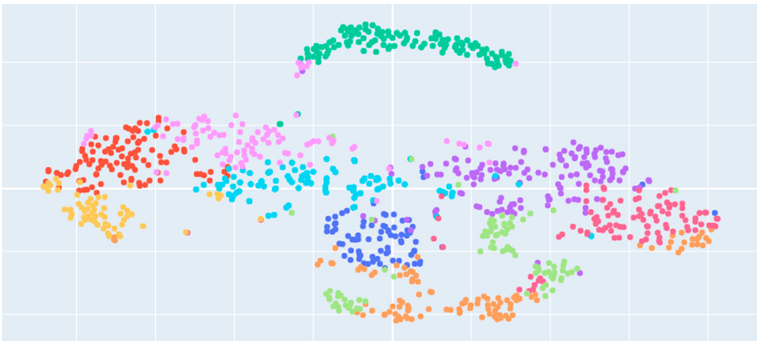

Fig. 6. Manhattan distances on feature layer

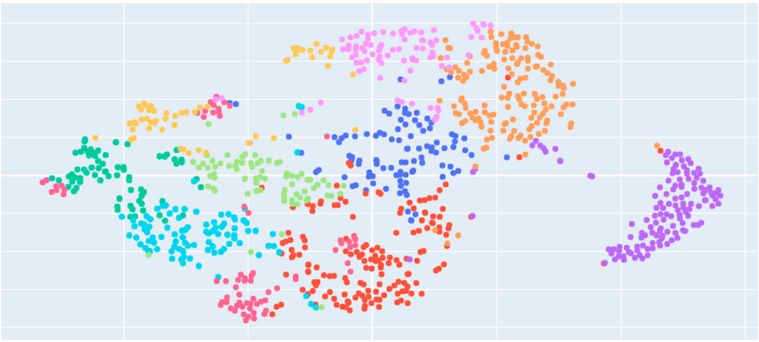

Fig. 7. Chebyshev distances on feature layer

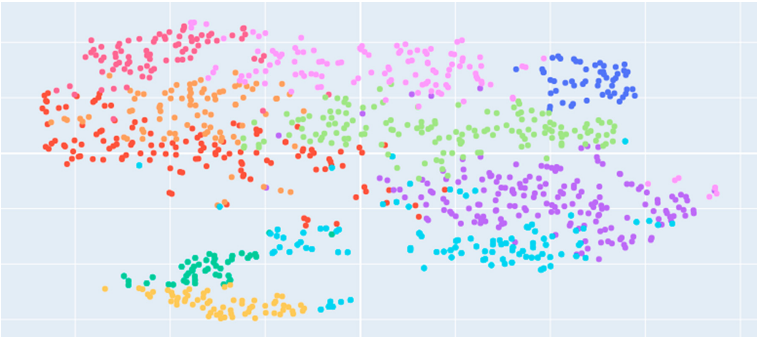

Fig. 8. Dot product on output layer

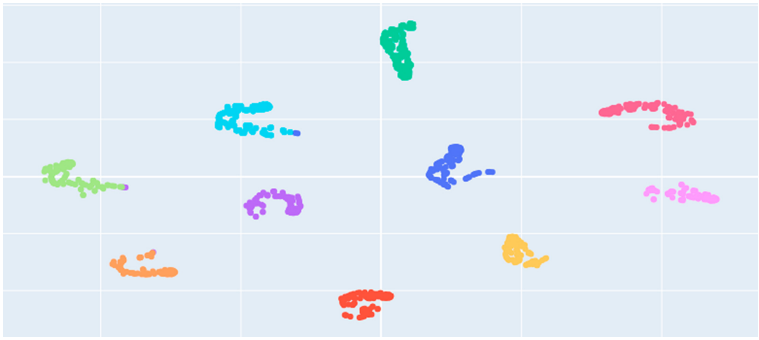

Fig. 9. Euclidean distances on output layer

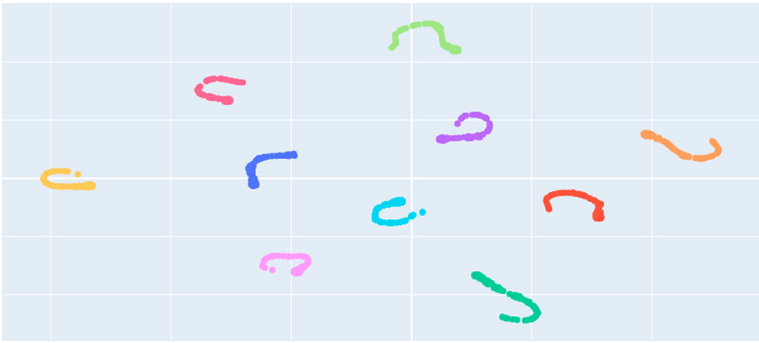

Fig. 10. Minkowski distances on output layer

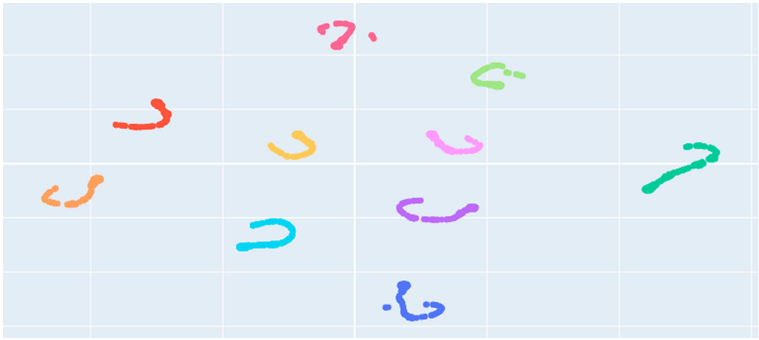

Fig. 11. Manhattan distances on output layer

Fig. 12. Chebyshev distances on output layer

4 Discussion and Conclusion

We can accredit some of the errors due to the fact that the quality of handwriting varies greatly depending on each person and sometimes it is unclear which digit was written. For example, someone's 9 might look like a 4. Even with this varied data set, the neural network was still able to cluster considerably accurately. We did some preliminary experiments a tougher data set that is harder to cluster. The data set is from the Canadian Institute For Advanced Research (CIFAR). We used the CIFAR-10 which are colored images that are 32×32 pixels. Since the images are colored in RGB, the image arrays have 3 times as many layers as the MNIST image arrays. The CIFAR-10 data set has 10 classes (categories). Some examples of the classes are cats, fish, trucks, etc. (Figs. 13 and 14).

Unlike MNIST, these data sets are not center oriented meaning the objects are not always in the center of the image, so it is harder to classify and cluster. Neural networks have low accuracy rates on CIFAR compared to MNIST. MNIST had an accuracy rate of 98% while the CIFAR-10 data set had an accuracy rate around 45%. A high accuracy rate means the model is able to correctly identify images which means it should be able to cluster them accurately as well. From the picture it seems that the output layer distances does cluster the data better than the distances from the feature layer. There is still significant muddling in the output layer which means out neural network isn't doing a good job. We used the exact same procedure we did on the MNIST data set. It would be interesting to do more experimentation on the CIFAR-10 data set and see if it follows a similar trend as the MNIST when other distances are applied. Perhaps we can try to improve the neural network and see how the clustering improves. It would also be beneficial to get quantitative measures on the clustering charts so we can do a deeper comparison. We also want to include optimization models to find the best metric for clustering. Our current ideas involve using Gurobi optimizer to minimize false positives and missed detection.

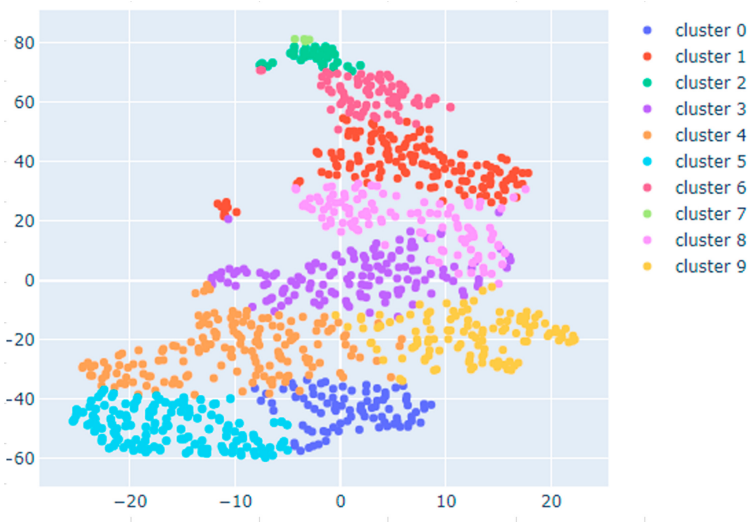

Fig. 13. Minkowski distances on flatten layer CIFAR10

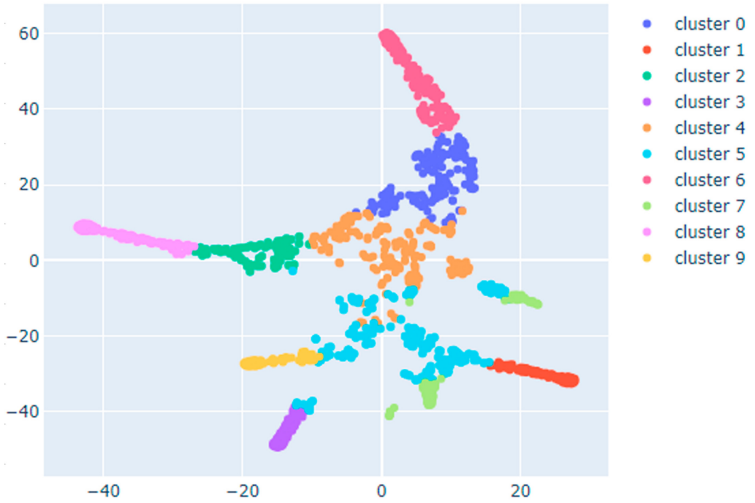

Fig. 14. Minkowski distances on output layer CIFAR10

References

1. Measuring distance or similarity. Packt Subscription
2. Likas, A., Vlassis, N., Verbeek, J.J.: The global k-means clustering algorithm. Pattern Recogn. **36**(2), 451–461 (2003). https://doi.org/10.1016/S0031-3203(02)00060-2. http://www.sciencedirect.com/science/article/pii/S0031320302000602. Biometrics

3. Mao, J., Jain, A.K.: Artificial neural networks for feature extraction and multivariate data projection. IEEE Trans. Neural Netw. **6**(2), 296–317 (1995)
4. Mao, Y., Yin, Z., Schober, J.: A deep convolutional neural network trained on representative samples for circulating tumor cell detection. In: 2016 IEEE Winter Conference, pp. 1–6. IEEE (2016)
5. Pradeep, J., Srinivasan, E., Himavathi, S.: Diagonal based feature extraction for handwritten character recognition system using neural network. In: 2011 3rd International Conference on Electronics, vol. 4, pp. 364–368. IEEE (2011)