Feature Extraction and Image Classification for 3D Data

Raja Mathanky Sankaranarayan
Computer Science Department
UCLA
mathanky04@g.ucla.edu

Shivam Dalmia
Computer Science Department
UCLA
sdalmia@ucla.edu

Vaishnavi Ravindran
Computer Science Department
UCLA
vaishnavi29@g.ucla.edu

***Abstract*—3D image classification is a growing need with autonomous systems; however, while 2D image classification has excellent accuracy already, 3D image classification still lags well behind. In this paper, we explore how existing classifiers may work on 3D image data and what changes and improvements are required and/or possible to achieve maximum accuracy. We especially focus on convolutional neural networks since they are state-of-the-art for 2D classification and note the changes and improvements possible herein.**

Keywords—3D, image classification, CNN, machine learning

I. Introduction

With the advent of robotics and autonomous systems that need to interact with their (dynamic) surroundings (such as surgical robots and self-driving cars), identification and classification of 3D objects becomes of paramount importance. While many applications of 2D image identification and classification exist, we have already reached excellent accuracy for the same; however, the application of the same models to 3D data do not yield the same results. Certain modifications are obviously required to prepare the 3D images for analysis using the existing models, but with the addition of features due to the new dimension, we may be able to optimize the algorithms themselves to yield better results.

The MNIST dataset [1] is often used as a starting point for image classification. In this dataset, we have 70,000 images, with a training set 6 times the size of the testing set. MNIST often serves as a benchmark dataset for various 2D classification algorithms.

Various machine learning algorithms have been tried on this dataset, and among these, Convolutional Neural Networks (CNN), a class of Deep Neural Networks have been the most successful, with a high accuracy of **99.79%** [2].

However, as aforementioned, we now want to explore the possibility of classifying 3D data, which is possible using a 3D version of MNIST [3]. This dataset is a set of point clouds generated from a section of the original MNIST database (around 10000 images). The point clouds have a mean of zero and a maximum dimension range of 1. The point clouds are generated using a script (provided with the dataset) which runs on the MNIST data.

In this project, we first prepare the 3D data for analysis using voxelization [4] and then use various well-known methods to classify the data. We gauge the performances of each of them and then attempt to use better feature extraction to improve the best of these. We also use 3D CNN [6] and compare it with 2D CNN for this purpose.

The remainder of this paper is structured as follows:

1. The next section contains a survey of the various papers that contributed to the methods and approach of this project.
2. The data to be used and the preparation required for the same follows.
3. We then list all the basic methods we use for classification.
4. We improve upon the best of these (2D CNN) by introducing 3D CNN and further improve this using Principal Component Analysis for the feature extraction.
5. Results for all the methods are then listed and compared.
6. Finally, we conclude and describe possible future work on this.

II. Literature Survey

A. *ImageNet Classification with Deep Convolutional Neural Networks [5]*

This 2012 NIPS paper uses Convolutional neural networks (CNNs) highly effectively (by implementing a large, highly optimized 2D convolution on a GPU) to classify a large ImageNet database. They also minimized overfitting and finally achieved excellent results on the subsets of ImageNet used in the ILSVRC-2010 and ILSVRC-2012 competitions.

B. *3D Convolutional Neural Networks for Human Action Recognition [6]*

This paper attempts to recognize human actions. The lack of availability of complex features derived from the input makes it require a novel approach. While Convolutional neural networks (CNNs) can automate feature construction by acting directly on the input, they are typically designed for 2D inputs. In this paper, the authors develop a novel 3D CNN model for action recognition. In fact, they attempt to extract features from four dimensions (space and time) by performing 3D convolutions, since motion information across several frames is required.

In 2D CNNs, 2D convolution is performed at the convolutional layers to extract features from local neighborhood on feature maps in the previous layer. Then an additive bias is applied, and the result is passed through a sigmoid function. For the purposes of

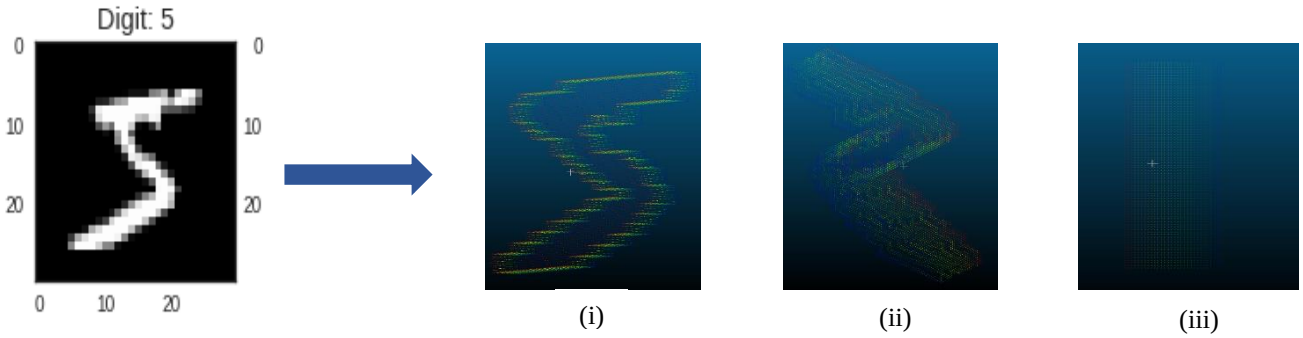

Fig. 1. 3D point clouds generated for the handwritten digit '5'. (i) Front view (ii) Back view (iii) Side view

this paper, a 3D convolution is performed in the convolutional layer of the neural network - this is achieved by "*convolving a 3D kernel to the cube formed by stacking multiple contiguous frames together"*. By this construction, the feature maps in the convolution layer is connected to multiple contiguous frames in the previous layer, thereby capturing motion information.

C. Classification Based Similarity Metric for 3D Image Retrieval [7]

This paper presents a method to obtain a weighted similarity metric for 3D image retrieval using Bayes decision theory. From an image, the most prominent features are selected by iteratively evaluating their efficacy in classifying images. Using combinatorial search techniques *cross-entropy* is minimized, resulting in a set of prominent features, which then serve as the index features for image retrieval. This feature-selection method, though meant for classification, serves well also for retrieval, according to the authors, the metric that does well at classifying the images will also do well in finding similar images, due to the semantic nature of both problems, and similar evaluation criteria. 3D grey-level volumetric images composed of normal and pathological neuroradiologic CT scans compose the dataset used for this research.

The authors use Kernel Regression (KR), a memory-based learning (MBL) technique to first classify the images using a certain set of features, and then iteratively improve the feature selection using Bayesian law, and then performing a combinatorial search over the space of features to get the best distance metric, and then minimizing an error metric known as cross-entropy.

This is another feature extraction method we could possibly use in the future to enhance the accuracy of the classifications.

D. Principal Componenet Analysis [8]

This paper introduces the concept of PCA which is a famous dimensionality reduction technique. The authors use a data matrix extract the dominant patterns in the matrix. They also look at outliers in the data which can be a major factor in the final accuracy.

The authors establish that PCA can be used for multivariate data analysis as it provides an approximation of the data. They conclude that it is however left to the responsibility of the data analyst to formulate the scientific issue at hand for the PC projections, PLS regression, etc.

The results depended on the scaling of the data matrix which is essential to be specified consequently. Variance scaling was used which is essential scaling each variable with a value of unit variance. The claim that this type of scaling can be used for any type of problem and if all the constant variables are left untouched.

III. Data and Preparation

The data for this project is the 3D version of the MNIST, from Kaggle [1] - the dataset was created by generating 3D point clouds from the images of the digits in the original 2D MNIST. It consists of 10000 images, split into 9000 training images and 1000 test images. Fig. 1 shows the point clouds generated for the handwritten digit '5'.

The Princeton ModelNet Challenge is the current benchmark for 3D object classification. To perform high-quality 3D object classification/recognition, there are two ways to represent the data: volumetric representation, where the 3D object is discretized spatially as binary voxels, i.e., the value is '1' if the voxel is occupied, '0' otherwise; or pixel representation, where the 3D data is represented as a set of 2D pixel images. While the leading submissions to ModelNet used CNNs on pixel representation, we aim to try volumetric representation with CNN and other techniques.

In the volumetric representation, we use 3D voxels, which is analogous to 2D pixels. Voxelization is the process of conversion of a geometric object from its continuous geometric representation into a set of voxels that best approximates the continuous object. [7]

We use this process to fit an axis-aligned bounding box called the **Voxel Grid** with a total of 4096 voxels, equivalent to the 4th level of an Octree. The figure below shows a histogram with the number of points present within each voxel for the 3D digit '5'.

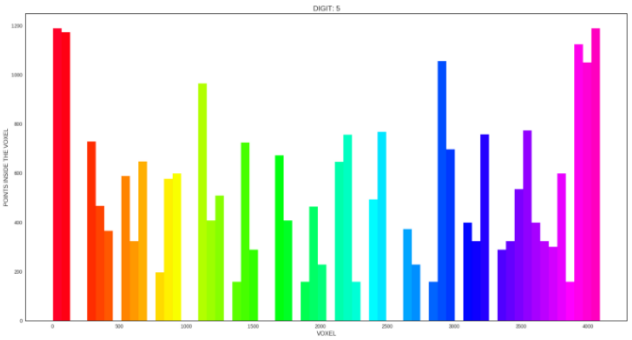

Fig. 2. Number of Points within each Voxel for sample digit 5

IV. Basic Methods

On our 3D data which is now voxelized, we run the following classifiers (from Python's `sckikit` library), and observe the results:

A. *Logistic regression:*

We perform logistic regression by minimizing the cost function below:

$$h = g(X\theta)$$
$$J(\theta) = \frac{1}{m} \cdot \left(-y^T \log(h) - (1-y)^T \log(1-h)\right)$$

However, the accuracy of logistic regression is very poor, and it serves only as a base to measure other classifiers.

B. *Decision trees*

We use decision trees to divide the input space into sections which can be appropriately labeled.

The decision tree aims to perform classification by designing a number of tests at each non-terminal node. At each node, it aims to reduce the purity of that node measured by the entropy. This is continued recursively at every level. For the entropy we can use either *Shannon Entropy* or the *Gini Index* given by, respectively:

Shannon Entropy:

$$i(t) = -\sum_{j=1}^{K} p(j|t) \log p(j|t)$$

Gini Index:

$$i(t) = 1 - \sum_{j=1}^{K} p(j|t)^2$$

These perform even worse than logistic regression (due to overfitting), but we can do better using random forests.

C. *K-Nearest Neighbors Classification*

For a given test image, k-NN finds the data-point (training image) closest to it (in terms of Euclidean distance) in the space; The test image is given the same label as this point. However, it performs poorly in this case as there is a lot of similarity between certain data points (e.g. {3,5,8} or {1,7})

D. *Random Forests*

Random forests are non-linear classifiers formed as an ensemble of decision trees. A shown in the figure below, they are a type of "bagging" algorithm. Random forests usually perform very well on multi-classification problems, which is our objective.

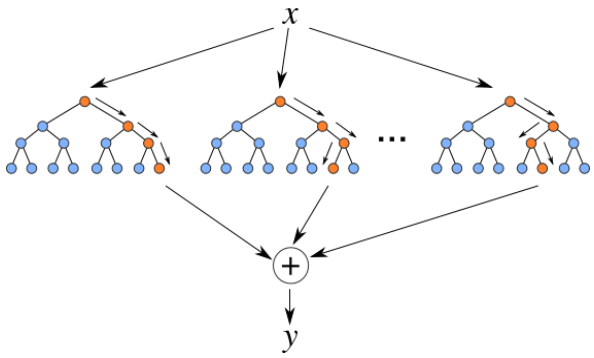

Fig. 3. Bootstrapping by means of a decision trees to form a random forest.

Random Forests overcome the problem of over-fitting in decision trees; and have the best accuracy among the basic methods, apart from 2D CNN.

E. *2D CNN*

In deep learning, a convolutional neural network performs multiple layers of perceptron learning and back propagation. They require minimal preprocessing compared to other algorithms.

It consists of an input and output layers with many hidden layers. The hidden layers of a CNN typically consist of convolutional layers, RELU layer i.e. activation function, pooling layers, fully connected layers and normalization layers.

For the 2D CNN, we use a 4-layered CNN in keras with two dense layers at the top. Each data point in our set is a 4096-D voxel, but we need to supply a 4D tensor to the CNN, of the form: `(batch_size, height, width, input_channels)`. We reshape our dataset into a 4D tensor: `(-1, 16, 16, 16)`.

F. *3D CNN:*

We first defined the model of the 3C CNN by defining the layers that make up the final model as follows:

- Convolution with 8 channels a kernel of 5 and stride of 2. 3D max pooling with LeakyRelu.
- Convolution with 16 channels a kernel of 3 and stride of 2. 3D max pooling with LeakyRelu.
- Convolution 32channels a kernel of 3 and stride of 2. 3D max pooling with LeakyRelu.
- Convolution with 64 channels a kernel of 3 and stride of 2. 3D max pooling with LeakyRelu.
- Flatten and dropout after every layer.

The hyperparameters tuned were the learning rate, number of epochs, batch size, activation layer, number of hidden layers and units. The optimizer used was RMSProp optimizer offered by Keras. This divides the gradient running an average over its recent magnitude.

The 3D CNN expects a 5d tensor of shape `(batch_size, conv_dim1, conv_dim2, conv_dim3, input_channels).` The input is reshaped into a 5D tensor of size `(-1, 16, 16, 16)` and feed it into our CNN. We finally use a tensor flow session to run under a tensor flow graph context. [9]

G. *PCA*:

Principal component analysis aims to perform dimensionality reduction by projecting the input data in a direction that has maximum variance of the data. This is a line passing through the mean of the data and the direction of the line is the top eigen vectors of the covariance matrix of the input data. We can visualize this as shown in Fig. 4. The loss function of PCA is the optimization of the following equation:

$$J_{d'}(m,a,e) = \sum_{k=1}^{n} \| (m + \sum_{i=1}^{d'} a_{ki} e_i) - x_k \|^2$$

Here, m is the mean of the data.
a_{ki} is the i^{th} principal component of the eigen vector.
e_i is the i^{th} eigen vector.

For our project, we use PCA to reduce the dimensions of our 3D data and we saw this increases the overall accuracy. It is important to note here that because this is a classification task, we should ensure PCA still keeps in tact the feature that are discriminative.

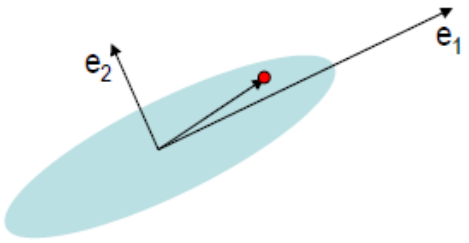

Fig. 4. Projection of data into the eigen space as shown by the 2 eigen vectors.

V. Methodology

The methodology followed for each of the algorithms involved the following outline:

1. **Data preparation, cleaning, and preprocessing**: Data cleaning in an important step in any Data Mining and Machine learning process. We preprocessed the data to ensure there was no out-of-range values, missing values, or impossible data combinations. We achieved this by normalizing our data given by using z-score normalization. This ensure the data lies within the range [-1,1] (other ranges like [0,1] can be chosen too). After normalization, the data underwent voxelization and finally we resized the data to fit an appropriately sized tensor. We also performed one-hot-encoding our labels. This was done using python's library `sklearn` which has a `OneHotEncoder` function.
2. **Data splitting:** Since we had the need to compare the accuracies of multiple algorithms, we had to split our dataset into train, validation, and test sets. We tried two approaches for splitting the data into these three sets – random subsampling and stratified subsampling. It is important to note that the test dataset should be left untouched and not used for the training set. This the common practice followed in Machine Learning.
3. **Training:** For each of the algorithms, we perform repeated training of the model until we can achieve the best accuracy for that model.
4. **K-fold cross validation.** Each of the algorithms used have several hyperparameters, which need to be properly hyper-tuned to achieve the best accuracy for that model. The hyper tuning for these parameters is done using k-fold cross validation. Here we split the training set into "K" folds. Then we use the hold-out-approach where we "hold-out" one out of the k folds for testing and use the remaining k-1 folds for training. This is performed K times and hence we arrive at K training accuracies and K testing accuracies. We then average these training accuracies and testing accuracies. This is performed for each of the combination of the hyperparameters for a given model. To finally compare their accuracies, we use the cross-validation set as the testing set to choose the best combination of the hyperparameters.
5. **Testing:** The test data set is used to get the final accuracies for each of the model. These are the accuracies we finally report in this project and use for the comparison of the results of the various algorithms.
6. **Visualization:** Due to the fact that we are working with 3D-data in this project, visualization of the data, results, and comparisons become an interesting and challenging aspect. We used external third-party 3D visualization software to this along with Python's `matplotlib` library.

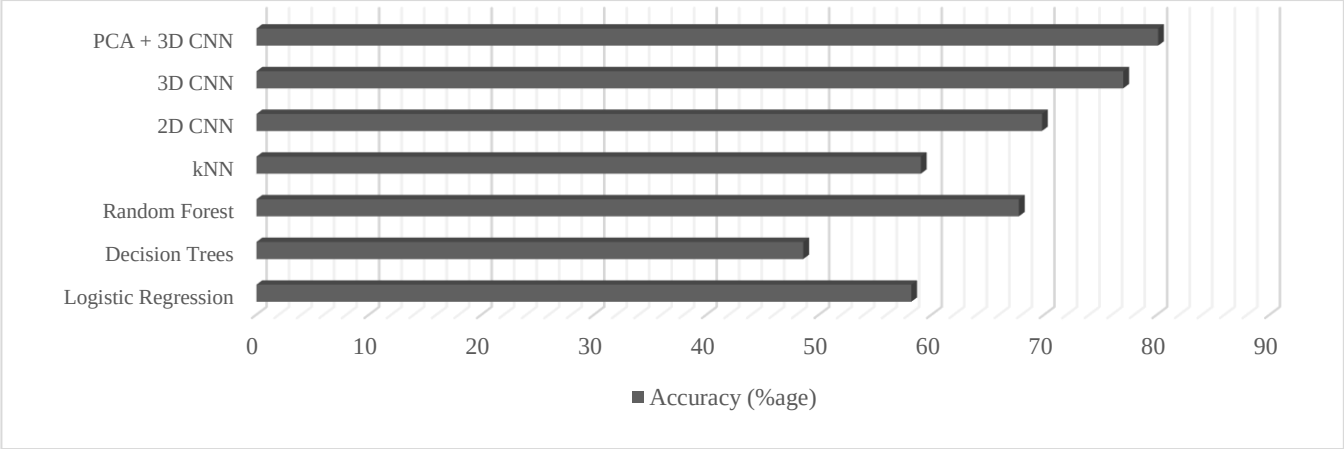

Fig. 5. Comparison of results

VI. Results

From the following table of accuracy values, Decision Trees have the lowest accuracy on the test dataset. This is because decision trees tend to overfit data easily, thereby reducing accuracy on unseen data. This problem is overcome by Random Forests, which help achieve a greater accuracy.

Method	Accuracy (%)
Logistic Regression	58.2
Decision Trees	48.6
k-NN	59.05
Random Forests	67.75
2D CNN	69.8
3D CNN	77.1
3D CNN with PCA	**80.1**

Logistic regression forms a linear boundary to classify the data and might misclassify digits such as '5' and '3' as '8'. This leads to a low accuracy. The case with K Nearest Neighbors is similar, as it computes the Euclidian Distance, which might not model the structure of the digits accurately.

CNN has a good accuracy because it models the structure of the data and takes edges and vertices into account. It uses multiple layers to model the features of the data. PCA with 3D-CNN achieves the best accuracy, as the PCA-transformed data is fed as input to the CNN, which extracts the most significant features that distinguish the images accurately.

VII. Conclusion And Future Work

In this project we performed image classification of 3D data, comparing various models and deciding which is optimal. Unsurprisingly, a neural-network model was the most accurate (as with 2D image classification), however, we needed a 3D neural network instead of a 2D one to achieve the best results.

Using the 3D CNN, we got a reasonably high accuracy, though it is nowhere close to the near-perfect accuracy of 2D classification. The addition of a third spatial dimension needs more sophisticated vision, and the supervised algorithm with limited test data is limited in its approach. The 3D CNN successfully captures the spatial data such as edges and vertices in the data. It extracts features at multiple levels using hidden layers, and this makes the model more discriminative.

However, we were able to achieve further improvement with Principal Component Analysis, which improved feature extraction by projecting the data on the PCA-transforms, thus achieving a good accuracy.

In the future, we can explore better feature selection, as outlined in [7], that improves the features iteratively. Another aspect to look at is higher convolution and more back-propagation in the 3D CNN so that the voxel identification is improved, and we obtain better accuracies.

References

[1] Castro, D. D. (2016, November 09). 3D MNIST. Retrieved from https://www.kaggle.com/daavoo/3d-mnist/data

[2] Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., & Fergus, R. (2013, February). Regularization of neural networks using dropconnect. In International Conference on Machine Learning (pp. 1058-1066).

[3] LeCun, Y., Cortes, C., & Burges, C. J. (2010). MNIST handwritten digit database. AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist, 2.

[4] Kaufman, A., Cohen, D., and Yagel, R. (1993) Volume Graphics. Computer 26, 7 (July 1993), 51-64.

[5] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[6] Ji, S., Xu, W., Yang, M., & Yu, K. (2013). 3D convolutional neural networks for human action recognition. IEEE transactions on pattern analysis and machine intelligence, 35(1), 221-231.

[7] Liu, Y., & Dellaert, F. (1998, June). A classification based similarity metric for 3D image retrieval. In cvpr (p. 800). IEEE.

[8] Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. Chemometrics and intelligent laboratory systems, 2(1-3), 37-52.

[9] Aggarwal, S. (2018, January 12). 3D-MNIST Image Classification – Shashwat Aggarwal – Medium. Retrieved from https://medium.com/shashwats-blog/3d-mnist-b922a3d07334.