On the Robustness of Multi-view Convolutional Neural Networks with Applications in 2D and 3D Image Classification

Soo Min Kwon, Ye Tao, Yu Wu*

Abstract

Traditional convolutional neural networks (CNN) have been widely adopted in different research areas to solve problems such as image classification. This technique, in general, aims to convolve different parts of the input data through a filter, to be later pooled and fed through different layers for prediction. Recent work in computer vision has shown that adding additional architectures to the classic CNN structure for 3D shape recognition improves performance, in the sense that using this new CNN framework dramatically increases prediction accuracy. In this paper, we investigate the robustness of this CNN framework, called multi-view CNN (MVCNN) in the settings of 2D and 3D image classification. Firstly, we empirically prove that changing the parameters of different MVCNN architectures for 2D renderings of 3D images for classification results in similar performance, showing that MVCNNs can be robust to different changes in its layers. Secondly, we also demonstrate the use case of MVCNNs to increase performance compared to traditional CNN methods on augmented, noisy 2D image datasets.

1 Introduction

Many modern applications involve using convolutional neural networks to solve problems regarding images, including cancer detection [1], [2], fMRI analysis [3], [4], and image classification [5], [6], [7]. Deep learning algorithms in general have had

*The authors are with the Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, New Brunswick, NJ 08854, `smk330@scarletmail.rutgers.edu`, `yt371@scarletmail.rutgers.edu`, `yw828@scarletmail.rutgers.edu`.

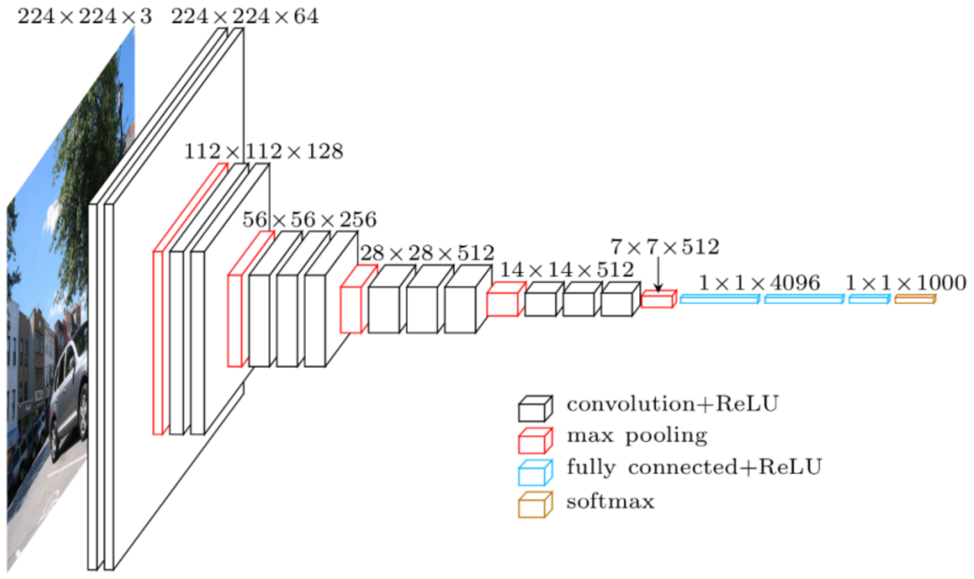

Figure 1: Example of a standard CNN architecture with different layers and a softmax layer for prediction. [8]

lots of attention in many fields, among which CNNs particularly have shown notable increase in performance when the data in question regarded images. Consider a standard CNN architecture as shown in Figure 1. The convolution layers located in the earlier stages extract "features" from the input data for further analysis. These features involve certain details in images, ranging from corners and edges to irregular shapes and objects. Intuitively, we can see that extracting these types of features for image analysis can be extremely beneficial. For example, Su et al. [7] exploits the benefits of using CNNs to solve the task of 3D shape recognition. Interestingly, they add an additional CNN portion to the standard framework to extract activation features of multiple views of a single image to feed forward into another CNN to make predictions. More specifically, the authors take 2D renderings of 3D shapes as the inputs to the MVCNN to train and classify the type of 3D shape that was tested. Surprised and inspired by the results of this paper, we take on the task to explore how we can leverage the MVCNN framework in various (mostly 2D) settings.

Our Contributions. Our objective in this paper is to explore how we can leverage the MVCNN framework to solve 2D and 3D image classification problems. In specific, we first follow the setup of Su et al. [7] to observe the robustness of the MVCNN model. We change certain parts of the architecture and observe its changes in performance. If the prediction accuracy is relatively high throughout our experiments, we can confirm that standard MVCNNs (i.e. MVCNNs with different number of layers and etc.) can be used efficiently for 3D image classification. To generalize, we also observe how the MVCNN model performs on 2D image datasets compared to a method which we call the SIFT+CNN$_2$ model. In essence, the SIFT+CNN$_2$ model mimics the structure of the MVCNN model, but with a different feature extraction step (e.g. SIFT). Details regarding this model will be explained in later sections. This comparison will prove that adopting the MVCNN model is efficient for both 2D and 3D image classification. To make 2D image classification a bit more interesting, we augment the data, described in the following sections. Our main contributions can be summarized as follows:

- Su et al. [7] has shown that using an MVCNN model for their setup dominates performance for 3D shape recognition. As hyperparameter tuning for MVCNN can be daunting, we demonstrate that using many different types of structures preserves the performance, showing the flexibility of the model.
- We translate the success of the MVCNN model on 3D image classification to 2D image classification. In specific, we take the MVCNN model approach and compare the prediction accuracy to a model we call SIFT+CNN$_2$.
- To make 2D classification a bit more interesting, we augment the 2D image dataset to make the task of classification a bit more challenging. Even in this scenario, we show that the MVCNN model outperforms *any* CNN method, including SIFT+CNN$_2$.

2 Background and Related Works

We dedicate this section to briefly inform the reader on convolutional neural networks, multi-view convolutional neural networks, and the main differences between the two. For a more comprehensive survey, we direct the reader to read the textbook by Goodfellow [9] and the paper by Su et al. [7].

Convolutional Neural Networks

Convolutional neural networks were introduced by LeCun in 1989 [10] as a neural network for processing data without structure (e.g. time-series data and images).

The most important operation (and sometimes known as the most confusing) of the CNN is the *convolution operation.* Mathematically, this operation, denoted by $*$, is

$$s(t) = (x * w)(t), \tag{1}$$

where $x(t)$ is referred to as the input, $w(t)$ is the kernel, and $s(t)$ is often called the feature map. With this, we can explain the general process of a convolutional neural network.

Convolution Layers. The main purpose of the convolution operation is to extract features from the input data, which in our case, are images. The benefits of this operation as opposed to other existing methods is that convolution preserves the spatial relationship between the pixels of the image. For example, traditionally, machine learning methods "vectorize" the data for the input, which destroys the spatial structure of the data. This would decrease accuracy compared to a method that keeps such a structure. Some terms important here are *depth* and *filter size*, and *filter strides.* Depth refers to the number of filters we use for the convolution operation. For example, because images are three-dimensional, we would use three filters stacked as a "feature map". The filter size simply refers to the dimensions of the filter. Filter strides, or just strides, is the step size of how much we move the filter per step. With these, we can define the convolutional layer step, which can be explained as a filter with some dimensions (generally smaller than the input size) and depth, that runs across an input image, convolving the pixels from the pixels of the previous filter to the next. The convolution layer also generally proceeds with an "ReLU" activation function, which can be read and learned in various resources [11], [12].

Pooling Step. The pooling step is rather simple. The pooling step, also referred to as downsampling, reduces the dimensions of the feature maps attained by the filters. The reason for having such an operation is intuitive: the more feature maps we have, the longer it make take the CNN to train. Of course, there may be a tradeoff in accuracy for downsampling such features, but not to such an extent, as observed in practice.

Fully-Connected Layer. The fully connected layer can be seen as the original layers of a standard neural network. In simple terms, these layers implies that the neurons in these layers correspond to every other neuron of the next layer. If the convolution layer is to learn special features about the input, the fully connected layer is to use the features learned to make a prediction. For a prediction task, the last layer of this is called a "softmax layer."

Putting these altogether, we can produce a simple convolutional neural network as shown in Figure 1.

Multi-view Convolutional Neural Networks

Multi-view CNNs can easily be seen as a “glorified” CNN. The MVCNN can solve image classification problems (and others), similar to problems that a standard CNN can solve. The steps are generally the same, however, with tweaks on how we obtain such features. As shown in Figure 2, the inputs to the MVCNN model are also images. Each image goes through a network called CNN_1, which consists of layers similar to that of the standard CNN. The purpose of CNN_1 is to extract features from the input data to be "view pooled" for CNN_2. In this architecture, CNN_2 performs predictions given the features. In specific, the objective of CNN_2 is to obtain a "compact shape descriptor", which in essence describes the labels of the data. One natural question here is that the standard CNN model explained in the previous step may not be directly compared to that of the MVCNN model. This is because the standard CNN model technically does not have a step to extract view based features. One way to get around this technicality is to use an operation such as Scale Invariant Feature Transform (SIFT) and a "bag-of-words" model which we describe and use in the upcoming sections, and name SIFT+CNN_2.

Related Works

Our work in this paper is to expand upon Su et al.'s work [7] on how multi-view CNNs can be used for 3D shape recognition, and further on 2D image classification. There is no directly related work on investigating the robustness of the MVCNN model, so we briefly discuss some of the works relevant to image classification.

Most related works for image classification investigate different types of descriptors or representations to observe changes in performance. For example, many works use "hand-crafted" features that contain certain shapes, distances, or angles of 3D models formed into bag-of-words models [13], [14], [15]. Intuitively, such a feature extraction method may seem to be efficient, however have some downsides. For example, 3D shape descriptors can be very high in dimensions when the input data is inherently high in dimensions (e.g. fMRI data). This can cause models to overfit due to the curse of dimensionality, where we have more parameters than the number of data samples available.

Other methods are related to using standard convolutional neural network methods for image classification. Such methods have similar approaches, which are to either use a simple CNN model, or to use feature extraction method for descriptors as inputs to a CNN model. Our analysis in this paper is to compare one of the proposed 3D methods (MVCNN) to these classic CNN methods.

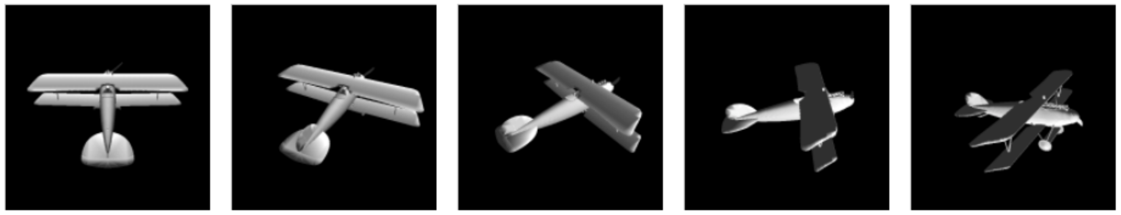

Figure 2: 2D renderings of 3D CAD models from the ModelNet40 dataset [7], [16]. Each view has the label 'airplane', which is used for both experiments.

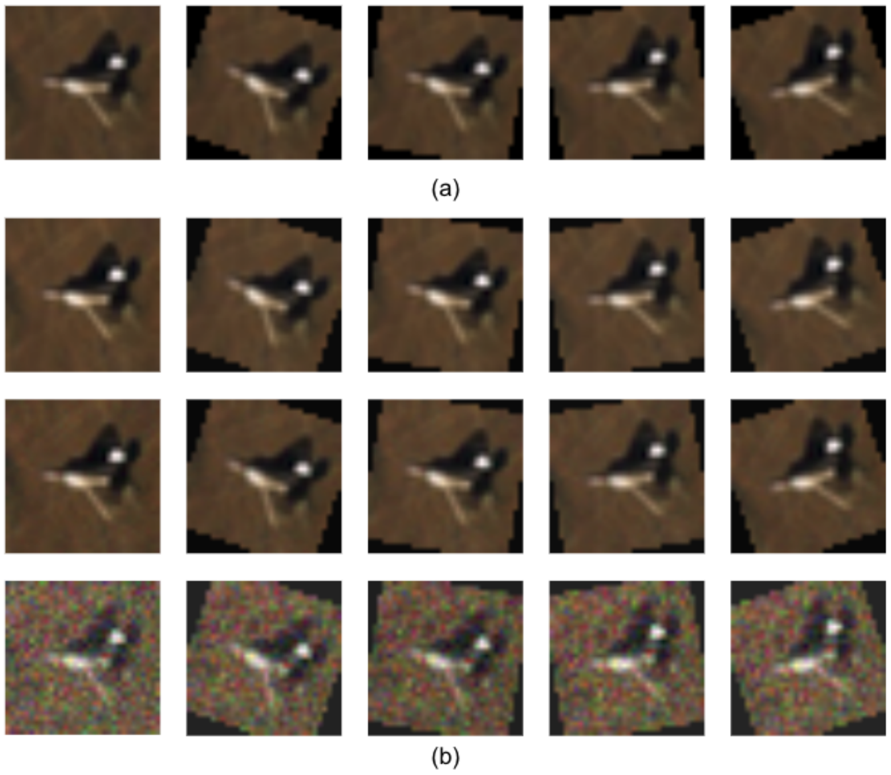

Figure 3: (a) Example of CIFAR-10 Part 1. From left to right, we can see the original image, followed by rotations of $25, 15, -25, -15$ degrees. (b) Example of CIFAR-10 Part 2. Row 1: Each image in (a) is added with noise $\mathcal{N}(0, 0.001)$. Row 2: Each image is added with noise $\mathcal{N}(0, 0.01)$. Row 3: Each image is added with noise $\mathcal{N}(0, 0.1)$.

3 Experiments

We base our experiments on two datasets:

1. ModelNet40[1]
2. CIFAR-10[2]

ModelNet40 Dataset

The ModelNet40 dataset [16] consists of a clean collection of 3D CAD models separated into 40 different classes. Some of these classes include (but are not limited to) airplane, bathtub, bookshelf, and dresser.
Pre-processing. Su et al. proposes that using 2D renderings of 3D images significantly increases prediction accuracy of 3D shape recognition. Thus, the ModelNet40 dataset is preprocessed to take different angles of the 3D CAD models originally provided. We can see some of these examples in Figure 2.

CIFAR-10 Dataset

The CIFAR-10 dataset [17] consists of $60,000$ images of 10 classes, where each image has dimensions $32 \times 32 \times 3$. There are a total of $6,000$ images per class, where the classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. There is a split of $50,000$ images for training and $10,000$ images for testing.
Pre-processing. The CIFAR-10 dataset is used to compare the performance between the MVCNN and the CNN with a feature extraction step, which we call SIFT+CNN$_2$. The main question we try to answer with this dataset is how the performance varies between the two models when the images are augmented and noisy. To answer this question, we modified (or pre-processed) the CIFAR-10 dataset in two ways, which we name CIFAR-10 Part 1 and CIFAR-10 Part 2.
CIFAR-10 Part 1. For CIFAR-10 Part 1, after normalization, we rotate each image in the dataset by the following degrees: $-25, -15, 0, 15, 25$. This "mimics" data augmentation/jittering as described by Su et al. Note that since the original dataset consists of $60,000$ images, this new dataset consists of $60,000 \times 5 = 300,000$ images. We hypothesize that the increase in sample size may also help overfitting in the training process.

[1]ModelNet40 is downloadable here: https://modelnet.cs.princeton.edu/
[2]CIFAR-10 is downloadable here: https://www.cs.toronto.edu/ kriz/cifar.html

CIFAR-10 Part 2. For CIFAR-10 Part 2, after normalization, we add Gaussian noise to each image in CIFAR-10 Part 1 with mean 0 and variance $0.1, 0.01, 0.001$. Note that this creates three different datasets, each separated by the variance added to the images. Each dataset consists of $60,000 \times 5 = 300,000$ images.

We show some examples of CIFAR-10 Part 1 and CIFAR-10 Part 2 in Figure 3.

3.1 3D Image Classification Experiment

For the first experiment, we follow the setup of Su et al. [7] to observe how the prediction accuracy changes as we vary the parameters (e.g. number of layers, type of pooling, number of dimensions in layers) of the MVCNN model on the ModelNet40 dataset. Note that there are two parts to the MVCNN: CNN_1 for feature extraction and view pooling and CNN_2 for prediction. Below, we present the original MVCNN model that Su et al. use, as well as the following changes that we make to observe prediction accuracy.

- **Original:** The original MVCNN was the VGG-M network, which consists of five convolutional layers, which we denote as $\text{conv}_{1,\ldots,5}$, followed by three fully connected layers, denoted as $\text{fc}_{6,\ldots,8}$. The penultimate layer of this model, fc_7 (which after ReLU non-linearity is 4096-dimensional) is used as image descriptor for CNN_2.

The following are the changes made for this experiment:

- 1^{st} **Modification:** The first modification was to add one layer to CNN_1, which makes six convolutional layers $\text{conv}_{1,\ldots,6}$ followed by three fully connected layers $\text{fc}_{7,\ldots,9}$.
- 2^{nd} **Modification:** The second modification was to delete one layer of CNN_1, which makes four convolutional layers $\text{conv}_{1,\ldots,4}$ followed by three fully connected layers $\text{fc}_{5,\ldots,7}$.
- 3^{rd} **Modification:** The third modification was to change the max view pooling layer (used in Su et al. setup) to average view pooling layer.
- 4^{th} **Modification:** The fourth modification was to add one layer to CNN_2, which makes five convolutional layers $\text{conv}_{1,\ldots,5}$ followed by four fully connected layers $\text{fc}_{6,\ldots,9}$.
- 5^{th} **Modification:** The fifth modification was to delete one layer of CNN_2, which makes five convolutional layers $\text{conv}_{1,\ldots,5}$ followed by two fully connected layers $\text{fc}_{6,7}$.

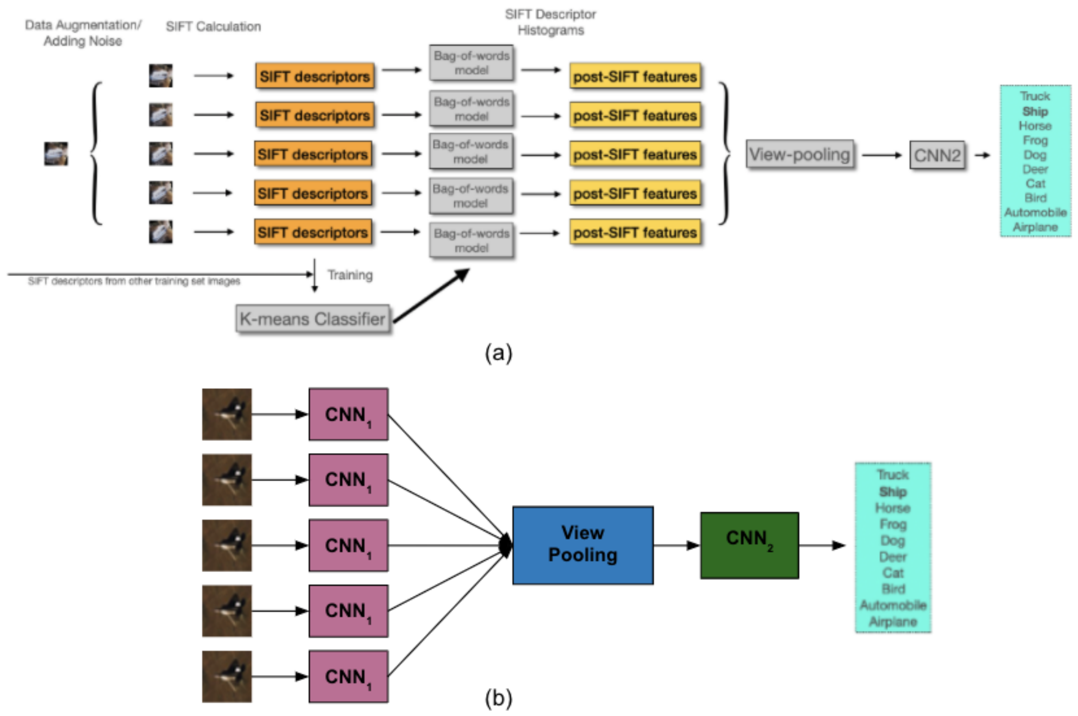

Figure 4: Visualization of models used in 3D and 2D classification experiments. (a) SIFT+CNN_2 model framework used mainly in 2D classification experiment. (b) MVCNN model used in both experiments, where we vary the parameters of CNN_1.

- 6^{th} **Modification:** The sixth modification was to change the parameters of the penultimate layer, which makes (after ReLU non-linearity) the layer 2048-dimensional to be used as an image descriptor.

The objective of these experiments is to show that even with these modifications, the MVCNN shows high accuracies which demonstrates the robustness of the model. The results of these experiments are shown in the following section.

3.2 2D Image Classification Experiment

To reiterate, the objective of the 2D image classification experiment is to show that the MVCNN model dominates the SIFT+CNN_2 in terms of prediction accuracy. For this experiment, we use all three datasets: ModelNet40, CIFAR-10 Part 1, and

CIFAR-10 Part 2. Below, we describe the two models that we train to compare the performance on the three datasets.

- **MVCNN:** The MVCNN model used for the 2D image comparison is similar to the one used in the reference paper [7]. That is, the different rotated (or augmented) images in both CIFAR-10 Part 1 and Part 2 "mimic" the 2D rendered images that the paper uses in their MVCNN model. Note that since the dimensions of the images of ModelNet40 and CIFAR-10 are different, parameters and layers in CNN_1 and CNN_2 are adjusted to fit the dimensions of the respective dataset. The CNN_1 portion of the MVCNN model architecture consists of four convolutional layers followed by three fully connected layers. The penultimate layer of this model which after ReLU non-linearity is 512-dimensional) is used as an image descriptor for CNN_2.

- **SIFT+CNN$_2$:** The SIFT+CNN_2 model that we use is a bit more complicated. Instead of feeding in the input images directly, we first take a feature extraction step, Scale Invariant Feature Transform (SIFT), as the inputs to the model. SIFT is a powerful feature extraction step, widely used for tasks such as image and object recognition. However, one problem that arises in using SIFT is that even though the dimensions of the images are the same, the number of keypoints detected by SIFT may vary. This results in having different dimensions for each sample in the input data. To resolve this issue, we adopt the bag-of-words model, which includes an array of "clustered" descriptors from SIFT. More specifically, we cluster all of the descriptors from the input image data into k clusters into an array which we call a "bag-of-words". This type of method is rather common, mitigating the issue of having different dimensions for the input data. This method now becomes comparable to that of the MVCNN, in that it shares CNN_2, but differs in SIFT and CNN_1. One thing to note is that many implementations of SIFT transforms image datasets into grayscale images before feature extraction. Thus, the features that were used for the bag-of-words model reflects this change.

We can visualize both of these models in Figure 4. In essence, the CNN method is a change in CNN_1, where the MVCNN uses CNN_1 as a means of feature extraction, and the CNN method uses the SIFT step.

4 Results

We dedicate this section to present the results of the two experiments described earlier.

4.1 3D Image Classification

For 3D image classification, since there were a total of six modifications of the original MVCNN used in Su et al.'s approach, there are six accuracies reported for each dataset. These results are tabulated in Table 1.

Table 1: Results for 3D Image Classification

Dataset	**Model**	**Prediction Accuracy**
ModelNet40	Original VGG-M Method	90.88%
	1st Modification	90.76%
	2nd Modification	**91.33**%
	3rd Modification	90.48%
	4th Modification	90.00%
	5th Modification	90.00%
	6th Modification	91.00%

From the modifications conducted in the 3D image classification experiment, we can observe that each modification returns a prediction accuracy that is similar to that of the original VGG-M method. With this, we conclude that the MVCNN model is robust to different types of changes, making hyperparameter tuning of layers and parameters a bit easier.

4.2 2D Image Classification

The 2D image classification experiment can be separated into four sub-experiments:

1. Comparison of MVCNN vs. SIFT+CNN_2 on **CIFAR-10 Part 1**.
2. Comparison of MVCNN vs. SIFT+CNN_2 on **CIFAR-10 Part 2**.

We tabulate these results in Table 2.

Table 2: Results for 2D Image Classification

Dataset	**Model**	**Prediction Accuracy**
CIFAR-10 Part 1	MVCNN	**83.57**%
	SIFT+CNN_2	26.00%
CIFAR-10 Part 2 (0.001)	MVCNN	**82.50**%
	SIFT+CNN_2	27.00%
CIFAR-10 Part 2 (0.01)	MVCNN	**82.81**%
	SIFT+CNN_2	26.20%
CIFAR-10 Part 2 (0.1)	MVCNN	**76.24**%
	SIFT+CNN_2	26.00%

Obviously, we can observe a trend of decreasing accuracies as the variance in added Gaussian noise increases. However, surprisingly, the results for the SIFT+CNN_2 model for 2D image classification are very poor. We have several plausible explanations for this phenomenon:

1. SIFT returns descriptors that are computed from a filter size of 16×16. Each image sample in the CIFAR-10 dataset, however, has dimensions 32×32, which can either cause zero padding or loss of information due to corners not being detected. In specific, if a keypoint was detected near the edges or corners of the image, the filter would cause zero padding, which would include extraneous, or even useless information in the descriptors. Further, if zero padding was not used, then the filter avoids keypoints near the edges, losing valuable information. Note that these issues depend on which implementation of SIFT is used.

2. In practice, it has been shown that the bag-of-words model used should have clusters proportional to 10 times the amount of classes available in the dataset ($10 \times 10 = 100$). However, due to the structural properties (e.g. dimensions, size) of CIFAR-10, we only have about 30 clusters, creating a very sparse matrix. Training a model with inputs that are sparse can cause models to overfit, returning undesirable results.

To prove that our points above were plausible, we tested the SIFT+CNN_2 model on the ModelNet40 dataset, which has image dimensions much larger than CIFAR-10. With this dataset, this model returned a prediction accuracy of 76.00%. With this, we hypothesize that the issue causing low performance may be in the structure of the training (and testing) data. Thus, the SIFT+CNN_2 model may only be effective for tasks regarding images/data of higher dimensions.

As our SIFT+CNN_2 returned poor results, we instead trained a standard CNN model consisting of four convolutional layers followed by three fully connected layers. One thing to note is that in this CNN model, we treat all the five augmented, noisy images as one object throughout the network. With this, we obtained accuracies tabulated in Table 3.

Table 3: Results for 2D Image Classification

Dataset	**Model**	**Prediction Accuracy**
CIFAR-10 Part 1	MVCNN	**83.57%**
	CNN	80.75%
CIFAR-10 Part 2 (0.001)	MVCNN	**82.50%**
	CNN	81.18%
CIFAR-10 Part 2 (0.01)	MVCNN	**82.81%**
	CNN	80.32%
CIFAR-10 Part 2 (0.1)	MVCNN	**76.24%**
	CNN	73.90%

For the CIFAR-10 dataset, we can conclude that the MVCNN is overall more favorable, however, with the standard CNN outperforming the SIFT+CNN_2 model.

5 Conclusion

Through this paper, we were able to show that the MVCNN model is robust and is most efficient to date in solving 2D and 3D image classification problems. We observed that the MVCNN is useful for 3D classification, when 2D renderings of the 3D models were taken as the input to the model. Further, we were also able to show that the MVCNN works more efficiently when the input data consisted of augmented, noisy images. Without pre-processing and/or quality control, noisy image datasets can be common, and we show that adopting such a method can help resolve this issue. To finalize, we summarize our main important discoveries below.

- The MVCNN model is robust to different changes throughout its architecture, mitigating some need for extensive hyperparameter tuning.
- The MVCNN model outperforms both the SIFT+CNN_2 model and the standard CNN model.

- The SIFT+CNN_2 model is not very useful when the dimensions of the input data are very small. We direct the reader to use such a model when the data dimensions are at least 100×100.

6 Reflections

In this section, we discuss some of the changes we made as well as respond to some of the peer reviews that we have received on our draft. From the review given by Dazhi Li, we added a couple sentences in the “Multi-view Convolutional Neural Network" section to say that MVCNN and CNNs can effectively solve the same problems. Of course, this would be with processing of the datasets to fit either architecture. Li addressed that there should be a stronger introduction on what the MVCNN can do, and we believe that our edits should suffice. In addition, Li also questions our motivation for proving the robustness of the MVCNN model. Our motivation simply revolves around us being surprised by the results of the paper by Su et al. [7], hoping to generalize the result to solve other types of problems (e.g. 2D image classification). Finally, the last comment made by Li was on using SIFT on grayscale images rather than RGB images. Actually, in fact, the SIFT implementation we used already transformed the images into grayscale images, and thus we write this in our SIFT+CNN_2 section. Additionally, it is worth noting that for many applications of image processing, color information does not help identify important edges, corners or other typical features. Thus, we believe this was not the issue for performance in our SIFT+CNN_2 model.

Zihao Ding makes a point suggesting that we should test our hypothesis of the poor results given by SIFT+CNN_2. The testing of this hypothesis was done by training the model on the ModelNet40 dataset, where the results were written after the hypotheses. We edited this paragraph to make our analysis stronger. The second point made by Ding was to run another experiment of the standard CNN vs. the MVCNN to show that the results were indeed impactful. This was completed using the ModelNet40 dataset, where the results were similar. However, we believe that a 3% increase is actually a notable change, as many researchers write papers that make this signficant change.

Lastly, the nitpicks made by both reviewers were also answered throughout the whole paper (after careful consideration).

References

[1] S. Hadush, Y. Girmay, A. Sinamo, and G. Hagos, “Breast cancer detection using convolutional neural networks,” 2020.

[2] Y. J. Tan, K. S. Sim, and F. F. Ting, “Breast cancer detection using convolutional neural networks for mammogram imaging system,” in *2017 International Conference on Robotics, Automation and Sciences (ICORAS)*, 2017, pp. 1–5.

[3] R. Zafar, A. S. Malik, A. N. Shuaibu, M. Javvad ur Rehman, and S. C. Dass, “Classification of fmri data using support vector machine and convolutional neural network,” in *2017 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, 2017, pp. 324–329.

[4] S. Sarraf and G. Tofighi, “Classification of alzheimer’s disease using fmri data and deep learning convolutional neural networks,” 2016.

[5] Y. Ioannou, D. Robertson, J. Shotton, R. Cipolla, and A. Criminisi, “Training cnns with low-rank filters for efficient image classification,” 2016.

[6] L. D. Nguyen, D. Lin, Z. Lin, and J. Cao, “Deep cnns for microscopic image classification by exploiting transfer learning and feature concatenation,” in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5.

[7] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, “Multi-view convolutional neural networks for 3d shape recognition,” in *Proc. ICCV*, 2015.

[8] J. Jordan, “Common architectures in convolutional neural networks,” https://www.jeremyjordan.me/convnet-architectures.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[10] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[11] J. Brownlee, “A gentle introduction to the rectified linear unit (relu),” https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[12] D. Liu, “A practical guide to relu,” https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7.

[13] S. B. Kang and B. K. P. Horn, *Extended Gaussian Image (EGI)*. Boston, MA: Springer US, 2014, pp. 275–278. [Online]. Available: https://doi.org/10.1007/978-0-387-31439-6_651

[14] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, “Rotation invariant spherical harmonic representation of 3D shape descriptors,” in *Symposium on Geometry Processing*, Jun. 2003.

[15] I. Kokkinos, M. Bronstein, R. Litman, and A. Bronstein, “Intrinsic shape context descriptors for deformable shapes,” *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 159–166, 2012.

[16] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao, “3d shapenets: A deep representation for volumetric shapes,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.

[17] A. Krizhevsky, “Learning multiple layers of features from tiny images,” Tech. Rep., 2009.

7 Temporary Section for updates/comments

Part 1: To show that MVCNN is robust enough.

(a) Original MVCNN: Use the VGG-M network from (add reference later) which consists of mainly five convolutional layers $\mathrm{conv}_{1,\ldots,5}$ followed by three fully connected layers $\mathrm{fc}_{6,\ldots,8}$. The penultimate layer fc_7 (after ReLU non-linearity, 4096-dimensional) is used as image descriptor. The accuracy of the classification is 90.88%.

(b) Modified MVCNN1 (Mainly change in CNN1): Add one layer to CNN1. MVCNN1 consists of mainly six convolutional layers $\mathrm{conv}_{1,\ldots,6}$ followed by three fully connected layers $\mathrm{fc}_{7,\ldots,9}$. The accuracy of the classification is 90.76%.

(c) Modified MVCNN2 (Mainly change in CNN1): Delete one layer of CNN1. MVCNN2 consists of mainly four convolutional layers $\mathrm{conv}_{1,\ldots,4}$ followed by three fully connected layers $\mathrm{fc}_{5,\ldots,7}$. The accuracy of the classification is 91.33%.

(d) Modified MVCNN3 (Mainly change in view pooling): The paper use max view pooling layer. We changed to use average view pooling layer. The accuracy of the classification is 90.48%.

(e) Modified MVCNN4 (Mainly change in CNN2): Add one layer to CNN2. MVCNN4 consists of mainly five convolutional layers $\mathrm{conv}_{1,\ldots,5}$ followed by four fully connected layers $\mathrm{fc}_{6,\ldots,9}$. The accuracy of the classification is 90.00%.

(f) Modified MVCNN5 (Mainly change in CNN2): Delete one layer of CNN2. MVCNN5 consists of mainly five convolutional layers $\mathrm{conv}_{1,\ldots,5}$ followed by two fully connected layers $\mathrm{fc}_{6,7}$. The accuracy of the classification is 90.00%.

(g) Modified MVCNN6 (Mainly change in CNN2): Change the parameters of CNN2. MVCNN6 consists of mainly five convolutional layers $\mathrm{conv}_{1,\ldots,5}$ followed by three fully connected layers $\mathrm{fc}_{6,\ldots,8}$. The penultimate layer fc_7 (after ReLU non-linearity, 2048-dimensional) is used as image descriptor. The accuracy of the classification is 91.00%.

Part 2: Using another dataset, i.e. Cifar10, to show the robustness of MVCNN. Parameters in CNN1 and CNN2 are adjusted based on the dimension of Cifar10 dataset.

(a) Augmented Cifar10 + MVCNN: accuracy 83.57%

(b) Noisy Cifar 10 (variance 0.001) + MVCNN: accuracy 82.50%

(c) Noisy Cifar 10 (variance 0.01) + MVCNN: accuracy 82.81%

(d) Noisy Cifar 10 (variance 0.1) + MVCNN: accuracy 76.24%

Part 3: Model description: "The CNN model consists of two parts, CNN1 and CNN2. To compare the results of CNN and MVCNN, we actually use the same constructions: four convolutional layers followed by three fully connected layers. The main difference between CNN and MVCNN is the view pooling layer. In the view pooling layer of MVCNN, we use element-wise maximum operation across the 2D renderings to aggregate the features selected by the CNN1, while in the CNN model, we treat all the five augmented and/or noisy images as one object, and then put them through the CNN1 to obtain the features per images, after that, we concatenate all the features and use CNN2 to do the classification based on the concatenated features. "

(a) Augmented Cifar10 + CNN: accuracy 80.75%

(b) Noisy Cifar 10 (variance 0.001) + CNN: accuracy 81.18%

(c) Noisy Cifar 10 (variance 0.01) + CNN: accuracy 80.32%

(d) Noisy Cifar 10 (variance 0.1) + CNN: accuracy 73.90%