Towards a Deeper Understanding of AdderNet

Kaifeng Zhang
2019012381
IIIS, Tsinghua University
zhangkf19@mails.tsinghua.edu.cn

Xingzhuo Guo
2019012383
IIIS, Tsinghua University
gxz19@mails.tsinghua.edu.cn

Abstract

This project is a research project on the CVPR 2020 paper AdderNet [2]. The paper introduced a fully-additive convolution network structure with competitive performance in many image classification tasks compared with conventional CNNs. However, the paper did not give a complete study on the intrinsic properties of AdderNet. In our work, we aim to obtain a deeper understanding of AdderNet, by focusing on the qualities of the features learned by the additive convolutional kernels. We performed several visualizations and adversarial attack techniques. The results show that the additive convolutional kernel indeed performs similarly to conventional convolutional kernels. However, there are also significant differences between them, causing some CNN-based techniques to become unavailable to AdderNet. Based on the observations, we discussed some possible structural extensions of AdderNet, including additive deconvolution and additive GAN.

1. Introduction

It is widely acknowledged that a multiplication operation costs more time than an addition or subtraction operation on a computer. However, almost all convolutional neural networks today use a multiplication-based kernel, causing a huge amount of multiplications for the forward pass. For this reason, researchers have long been exploring different architectures for neural networks to reduce the number of multiplication operations. Significant achievements in this area include BinaryConnect [3] and BNN [4]. They transform the weight parameters in neural networks from real numbers into binary numbers, and the activate functions are binarized as well. However, although there have been many further works on BNN, [11, 1, 18], it still suffers from a large performance gap compared with classical convolutional neural networks.

The invention of AdderNet has brought a great change to this field. Different from previous works that aim to compress the model by precision, AdderNet achieves this by proposing an entire new operation, the additive convolution, that effectively replaced all the multiplications in conventional CNN into additions and substractions. There are many follow-up works on AdderNet, including AdderSR [13], ShiftAddNet [16], PKKD-ANN [15].

However, while making a huge impact, the paper also received some controversy, which mainly focused on the efficiency, the training tricks, and model capability. Many people doubt whether AdderNet can truly make a difference in computational cost since the network runs no faster than a CNN on GPU while requiring a larger memory cost. However, that has nothing to do with AdderNet itself; the problem is that modern GPUs are specially designed for matrix computations. The potential of AdderNet will be revealed as long as addition-specific acceleration methods are invented. Many papers are working on this goal [7, 14].

Our project will not focus on improving the running speed of AdderNet. Instead, our goal is to focus on the idea of additive convolution itself. In the original paper, the authors visualized the features that AdderNet learned and claimed that the clustering of feature in AdderNet is very reasonable. However, in our perspective, this is not convincing enough. Many intrinsic properties of the additive convolution are yet to be discovered.

Studying how CNN works has been a large research field. The main topics in this field include visualizing CNN features, attacking CNN, etc. In our work, we aim to apply these techniques to the AdderNet as well. This will give us a further understanding of AdderNet.

2. Related Works

2.1. AdderNet

In this section we will introduce the detailed mathematical formation of AdderNet.

Convolution Convolutions in classical CNNs computes the cross-correlation between the feature maps and the convolutional kernels. AdderNet inherits the idea of cross-correlation, but changed the measure of correlation from the dot product between the image and filter into their ℓ_1-distance. Consider an input feature $X \in \mathbb{R}^{H \times W \times c_{in}}$, and a filter $F \in \mathbb{R}^{d \times d \times c_{in} \times c_{out}}$, where H, W, c_{in} are the height, width, number of channels of the input feature, d is the kernel size and c_{out} is the channels of the output feature. Then the convolution result Y can be abstracted as

$$Y_{m,n,t} = \sum_{i=0}^{d-1} \sum_{j=0}^{d-1} \sum_{k=0}^{c_{in}-1} S\big(X_{m+i,n+j,k}, F_{i,j,k,t}\big),$$

where $S(\cdot,\cdot)$ is any binary function. In classical convolution, we take S as multiplication, i.e., $S(x,y) = x \times y$. In AdderNet, S is taken by $S(x,y) = -|x-y|$, i.e.,

$$Y_{m,n,t} = -\sum_{i=0}^{d-1} \sum_{j=0}^{d-1} \sum_{k=0}^{c_{in}-1} \big|X_{m+i,n+j,k} - F_{i,j,k,t}\big|.$$

In this case $Y_{m,n,t}$ measures the negative ℓ_1-distance of the two patches, and in a way represents the cross-correlation.

Hard Gradient After applying Adder layer, the partial gradient with Y with respect to F is

$$\frac{\partial Y_{m,n,t}}{\partial F_{i,j,k,t}} = \text{sgn}\big(X_{m+i,n+j,k} - F_{i,j,k,t}\big),$$

which only takes the value ± 1 or 0. However, such gradient ignores the influence of the difference in value and leads to unstable training and bad results. Therefore, the paper introduces HardTanH function

$$\text{HT}(x) = \begin{cases} x & \text{if} \quad -1 \le x \le 1 \\ 1 & \qquad\qquad x > 1 \\ -1 & \qquad\qquad x < -1 \end{cases}$$

and redefines the partial gradient as

$$\left(\frac{\partial Y_{m,n,t}}{\partial F_{i,j,k,t}}\right)_{redefined} = \text{HT}\big(X_{m+i,n+j,k} - F_{i,j,k,t}\big).$$

Adaptive Learning Rate The redefined gradient is still hard on filters, so it is likely to cause gradient explosion. The paper applies adaptive learning rate to scale the update amount in different layers to be the same. Formally, suppose $\Delta L(F_l)$ is the gradient of the filter in layer l, then the adaptive learning rate is defined as

$$\alpha_l = \frac{\eta\sqrt{k}}{\|\Delta L(F_l)\|_2},$$

where η is a global hyper-parameter to control the amount of learning rate and k is the number of parameters in F_l.

2.2. Visualization in CNN

Visualizing convolutional filters in CNN means finding the input pattern that maximizes a certain filter's activation. To achieve this, there are different approaches.

Activation optimization A simple idea is to just find an image in the dataset that maximizes the filter's output value. A more advanced idea is to start from a random image, then try to optimize the pixel values. This is proposed in the year 2009 by [5]. Later, [12] improved this method by using gradient descent as the optimization approach.

DeConvNet Another type of approach is known as the DeConvNet method, first proposed by [17]. The method "inverts" a CNN to find which image pixels are most correlated with the activation of a filter. The convolutional layers are replaced by deconvolutional layers with same weights; the max-pooling layers are replaced by max-unpooling layers with a pooling map that records where the maximum values are before the pooling. Then the sequence of the layers is reversed. We feed the activation of a specific layer into the network, then the output will be the visualized features.

2.3. Adversarial Attacks

Researchers have shown that adding a tiny noise to an input image may cause the network to misunderstand the input and give a wrong classification. Attacking methods include white box attacks and black box attacks. White box attacks calculate the derivative of the loss with respect to the input, then add a tiny noise along the gradient direction to maximize the loss. Black box attacks do not rely on the details of the network or the gradient. Instead, they approximate the loss and the gradient from the inputs and labels and use the approximated gradient to attack.

Fast Gradient Sign Method (FGSM) FGSM [6] is the first work of adversarial attack. It measures the impact of a tiny noise on image quality by its ℓ_∞ norm, i.e., the maximum intensity among noise pixels. With bound ϵ as the maximum ℓ_∞-norm of noise, FGSM calculates the sign of the gradient of the input image, and obtain the adversarial data by

$$X^{adv} = X + \epsilon \cdot \text{sgn}\big(\nabla_X \mathcal{L}(X, y)\big),$$

where X, y stand for the input and the label, and $\mathcal{L}$ is the loss function.

Projected Gradient Descent (PGD) PGD [8] applies the gradient descent method on an input image. Since there is a bound of the norm of noise, each iteration it projects the data to the surface of the hyperball with radius ϵ if the norm exceeds the bound. Since the gradient is too small, researchers use the sign of gradient instead. The whole process can be expressed as:

$$X^{(0)} = X,$$
$$X^{(t+1)} = \mathrm{proj}_{X,\epsilon}\Big(X^{(t)} + \alpha \cdot \mathrm{sgn}\big(\nabla_{X^{(t)}} \mathcal{L}(X^{(t)}, y)\big)\Big),$$

where α is learning rate, a hyperparameter, and proj is the projection function. Specifically, if we take ℓ_∞-norm bound, then the projection is a simple clipping.

Black Box Attacks Most of the time, the gradient to the input image and the model structure is unknown to attackers. All an attacker can do is to input an image and get the label. Researches [9] have proposed that we can train a local deep model with input images and corresponding labels given by the black model, then generate attack data with local model to attack the black model.

3. Approaches

3.1. Visualizing AdderNet

Gradient-based approach The basic idea of gradient-based feature visualizing is to perform a fixed iteration of gradient descent on the input image pixels, with the target function be the mean activation of a certain filter. Usually, the visualization effect depends on the initialization of the input image. If the input image is composed of random pixels, then the feature units will be randomly distributed across the entire image. If the input image is a picture, then the features will act as an attention map of the specific filter on the picture. For instance, if the input is a picture of a cat, and the filter recognizes cat ears, then the ear of the cat in the picture will have large gradient compared with other regions.

We adopted some useful tricks for better visualization quality. When performing random initialization, we start from a relatively small random image, and alternately perform gradient descent and upscaling. This trick can reduce the number of repeated feature units in the final visualization result. We also adjust the learning rate depending on the model structure. Typically, for AdderNet, the norm of the gradient is usually small due to the hard gradient clipping. Hence we use a $10\times$ larger learning rate compared to CNNs.

Deconvolutional Approach We also tried to adopt the deconvolutional method to visualize AdderNet. In conventional CNN, we only need to transpose the filter to turn a convolutional layer into a deconvolutional layer which will approximately recover the input. However, AdderNet does not have an appropriate definition of its inverse.

We propose a "deconvolutional" layer as follows: First, we transpose the filter. Then, we compute the difference of the filter and corresponding map. Suppose the input feature map is Y, and we want to compute X' from Y and F to recover X. Then we let

$$X'_{i,j} = \sum_{p=0}^{d-1}\sum_{q=0}^{d-1}\big(-Y_{i-p,j-q} + F_{p,q}\big).$$

By plugging in Y as the convolution of X and F, we have

$$X'_{i,j} = \sum_{p,q=0}^{d-1}\Bigg(\sum_{p',q'=0}^{d-1} |X_{i-p+p',j-q+q'} - F_{p',q'}| + F_{p,q}\Bigg).$$

If the original X is similar with the filter F (i.e. Y has small norm), then the recovered result X' will also be similar with F. Note that we could have taken an absolute value on X', resulting in

$$X'_{i,j} = \sum_{p=0}^{d-1}\sum_{q=0}^{d-1}\big|Y_{i-p,j-q} - F_{p,q}\big|$$

which is similar with the original definition of additive convolution, but that would make X' all positive, which will probably cause some problems.

Following this definition, we applied the DeConvNet visualization framework onto the AdderNet. The results will be shown and discussed in the next section.

3.2. Adversarial Attacks on AdderNet

White box attack We respectively use the two white box attacking methods, FGSM [6] and PGD [8] on AdderNet, to explore the robustness of AdderNet.

Black box attack The method of black box attack requires us to train a local deep model with the output of AdderNet. So we train a convolution-based network to imitate the Adder-based network. We take the images from dataset as the input of the network, and the output of AdderNet as labels. Then we apply white box attack methods on this model to test the performance of black box attack of AdderNet. As a control group, we also train two convolutional-based networks in a similar way.

3.3. Transposed Adder Layer and Additive GAN

We further extend AdderNet to transposed convolutional layers and try to achieve an Additive GAN. We take the structure of DCGAN [10], but replace the convolutional layers in a discriminator with Adder layers, and replace the transposed convolutional layers in a generator with our transposed Adder layers. We apply zero-padding for the upsampling process in transposed Adder layers. Since the output of AdderNet is always negative, it is impossible to remove normalization layers in a generator. We use instance normalization in the generator, and use batch normalization in the discriminator.

4. Experiment Results

4.1. AdderNet Models

For visualization and adversarial attack, we adopted 2 popular CNN architectures, ResNet and VGG, as the base architecture for our AdderNet. The ResNet version contains 20 layers with 9 residual blocks; the VGG version contains 16 convolutional layers, and batch normalization is used after each layer. (Batch Normalization is a must for AdderNet). The results of the AdderNet will also be compared to the ImageNet-pretrained ResNet18, VGG16 and VGG16_bn.

We used CIFAR-10 dataset to train our AdderNet model. We used SGD optimizer with learning rate 0.1, momentum 0.9, and cosine learning rate decay. The learning takes 400 epochs in total. ResNet20 AdderNet successfully reached a 91.7% accuracy on validation set. The training curves are shown in Fig.1. For convenience, the VGG16 AdderNet is only trained for around 160 epochs, resulting in a validation accuracy of over 80%.

4.2. Gradient-based Visualization

The gradient-based visualization is performed on all models. We used the SGD optimizer, with learning rate 0.2 for CNN and 2 for AdderNet. We performed 50 iterations of gradient descent before each upscaling step. We used upscaling ratio 1.4 and 6 upscaling steps. The results are shown in Fig.2.

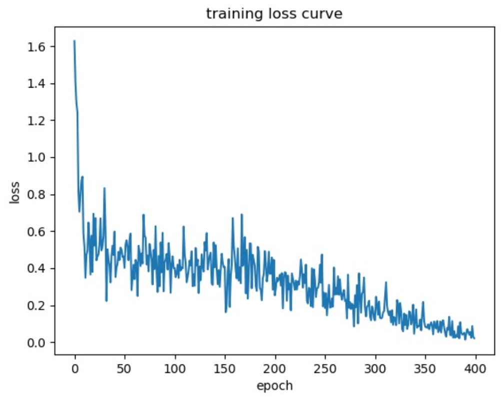
We also performed the experiment of taking a picture as input, and visualizing the attention pattern of a specific filter on the picture. The result is shown in Fig.3.

4.3. Deconvolution-based Visualization

In the deconvolution-based visualization, we adopted a deconvolutional network that inversely recovers the input image from the feature map. We set all but the maximum activation value to 0 in the feature map, so that the recovered region is the receptive field of the maximum activation point of that filter.

The design of the network is similar with that of the paper [17], with 2 main modifications: the first one is that we implemented our own deconvolution for the additive convolutional layer; the second one is that we added the inverse of the batch normalization layer into the pipeline, since the original paper did not experiment on models with batch normalization.

The results are shown in Fig.4. Note that the results on AdderNet might not appear to be successful. We suspect



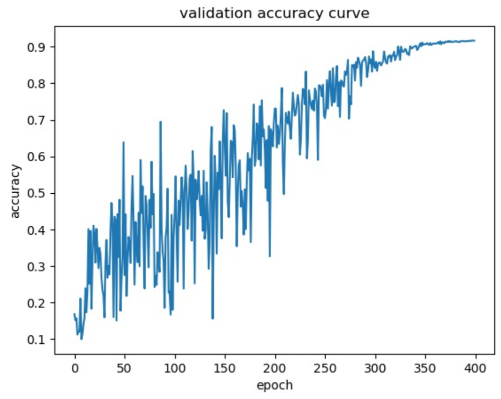


Figure 1. The training loss curve (top) and the validation accuracy curve (bottom) of the ResNet20 AdderNet.

that the cause is the batch normalization layer instead of the additive layer. In all of our experiments, DeConvNet behaves poorly on models with batch normalization.

4.4. White box attack

We respectively tried FGSM [6] and PGD [8] on Adder and convolutional based ResNet20 and CIFAR010 dataset with $\epsilon = 2, 4, 8, 16$. The result is shown in Table.1. In this experiment, the Adder-based ResNet performs better than the convolutional-based ResNet in all aspects, but it still does not show very strong robustness against white box attacks.

4.5. Black box attack

We have a convolutional ResNet20 as the local model. We still use CIFAR10 dataset as input images, but use the output of the Adder-based network as labels. For the control, we train a convolutional ResNet32 as the black model and another convolutional ResNet20 to as the local model to imitate that ResNet32. The result is shown in Table.2. One interesting fact is that, the black-box attack from a convolutional-based network to an Adder-based network does not seem that effective as the attack between two

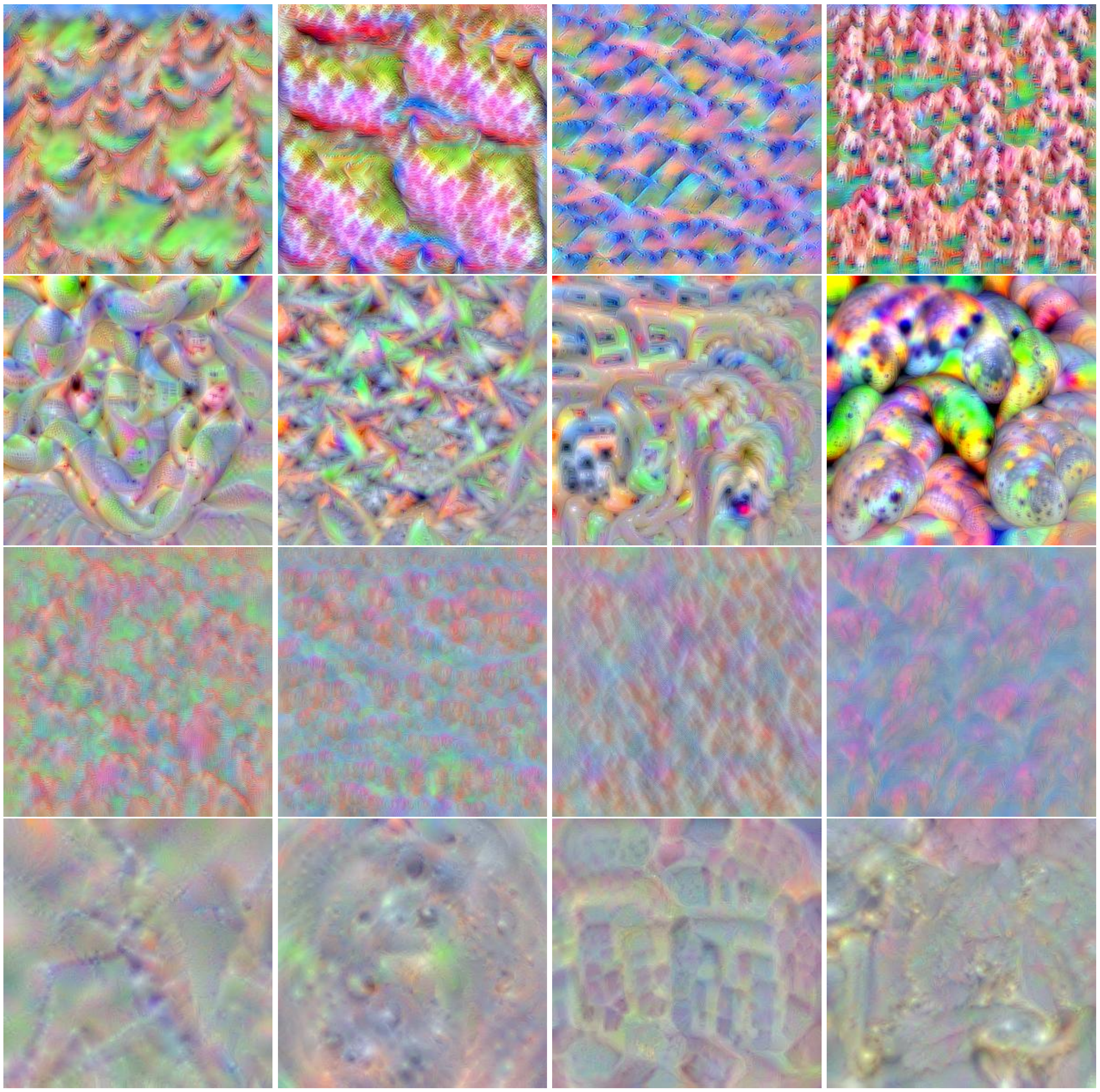


Figure 2. The gradient-based visualization results. From top to bottom, the 4 rows are: VGG16 AdderNet, pretrained VGG16, ResNet20 AdderNet, pretrained ResNet20. The features are all from the last convolutional layer. As we can see, both AdderNet and conventional CNN capture some unique high-level features, e.g. the shape of animals, or complex texture patterns. Compared with CNN, AdderNet has shown a more regular permutation of feature units, which is a desired property. Compared with VGG, The features of ResNet appears to be paler. This might be due to the effect of the shortcut connection.

convolutional-based networks. In other words, an AdderNet is more robust than a convolutional network against the black-box attack from another convolutional network. This is most evident in the accuracy of PGD, which always overfits the local model. It implies that, the non-robust feature extracted by an AdderNet is not exactly the same as that of a convolutional network. This explains their differences in the visualization.

4.6. Additive GAN

We have trained an Additive GAN on MNIST dataset. We follow the model of DCGAN [10], and use a fully con-

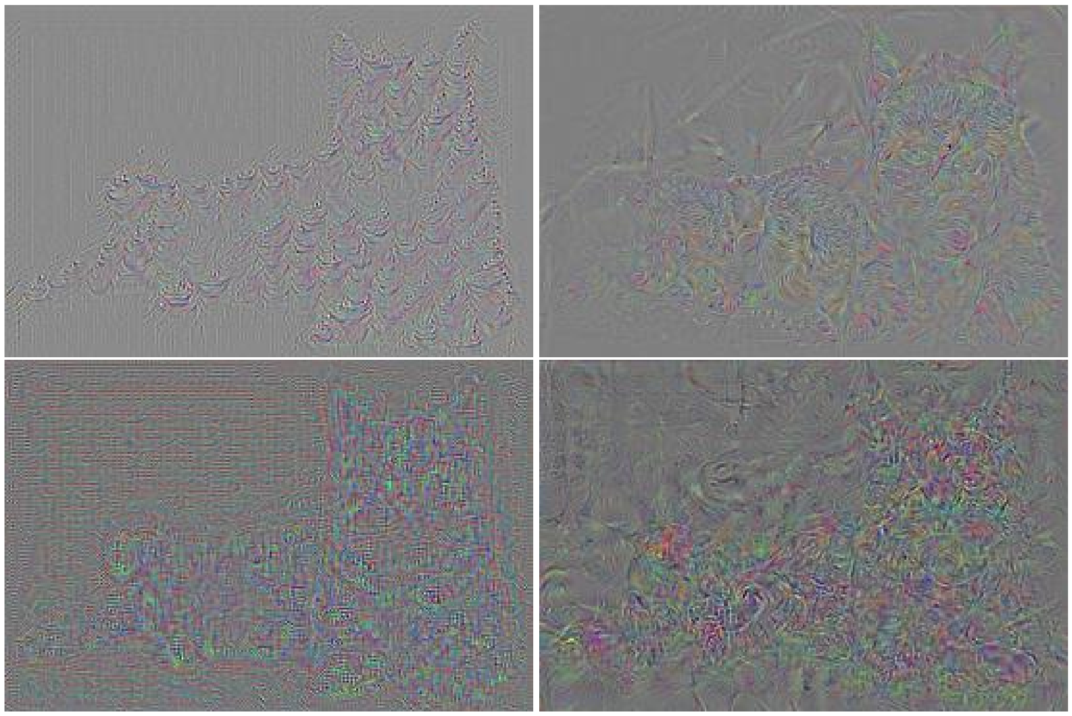


Figure 3. The gradient-based visualization result of the input picture. The first row: VGG16 AdderNet, VGG16; the second row: ResNet20 AdderNet, ResNet20. As we can see, the gradient focus on features of cat, including ears, eyes, fur, etc. Compared with CNNs, The gradient map of AdderNet is "cleaner" and more regular. This is a desired property since it will make the network learn only robust features.

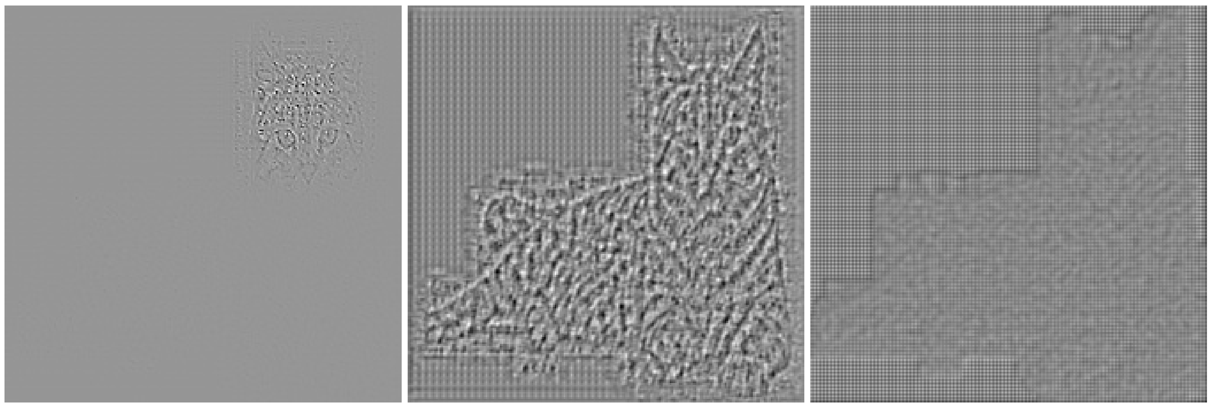


Figure 4. The deconvolution-based visualization result. From left to right: pretrained VGG16, VGG16 with BN, VGG16 AdderNet. The VGG16 visualization is successful. The filter captures the cat face as the key feature. For VGG16 with BN and AdderNet, the visualization result covers the entire cat body. One possible explanation is that batch normalization correlates different features.

volutional network. We use a 10×0.0005 learning rate and batch size 64. We use normalization as stated in 3.3. In this task, however, the network totally collapses, as shown in Fig.5. The output of the discriminator converges to 0.5 and cannot be further trained. Additive GAN still needs more exploration.

5. Conclusion

In this project, we have explored the intrinsics of the AdderNet model by various visualization and adversarial attack techniques.

The visualizations confirmed the claims in the original AdderNet paper that AdderNet is able to capture reasonable features. The gradient properties of AdderNet and CNN are similar in many aspects, which makes the visualization methods designed for CNNs also applicable to AdderNet.

	Adder	Convolutional
FGSM, $\epsilon = 2$	45.46%	33.23%
FGSM, $\epsilon = 4$	36.95%	26.96%
FGSM, $\epsilon = 8$	28.88%	20.86%
FGSM, $\epsilon = 16$	21.67%	15.45%
PGD, $\epsilon = 2$, $n = 5$	35.40%	27.50%
PGD, $\epsilon = 4$, $n = 10$	8.39%	4.81%
PGD, $\epsilon = 8$, $n = 20$	0.34%	0.13%
PGD, $\epsilon = 16$, $n = 40$	0.00%	0.00%

Table 1. Accuracy of Adder-based and convolutional-based ResNet against FGSM and PGD. Here ϵ stands for the bound of ℓ_∞ norm or noise, and n stands for the number of iterations.

	Conv-20 to Adder-20	Conv-20 to Conv-32
FGSM, $\epsilon = 2$	54.26%	45.60%
FGSM, $\epsilon = 4$	47.16%	38.66%
FGSM, $\epsilon = 8$	36.32%	29.42%
FGSM, $\epsilon = 16$	24.62%	20.36%
PGD, $\epsilon = 2$, $n = 5$	84.90%	82.60%
PGD, $\epsilon = 4$, $n = 10$	75.38%	68.84%
PGD, $\epsilon = 8$, $n = 20$	57.54%	42.00%
PGD, $\epsilon = 16$, $n = 40$	36.32%	15.48%

Table 2. Accuracy of black box attack.

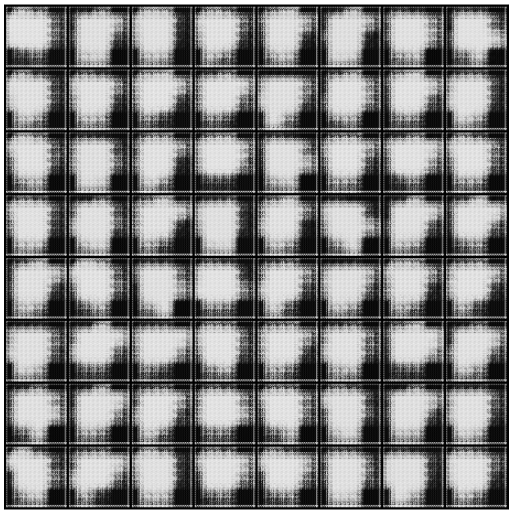


Figure 5. Collapse of Additive GAN.

However, they are not completely identical. Firstly, AdderNet has the drawback of not being able to construct a deconvolutional layer. AdderNet is also highly dependent on the batch normalization layer. The magnitudes of the gradient flow of AdderNet and CNN are also dissimilar.

The adversarial attack experiments show that AdderNet is more robust against many classical white-box attacking methods, including PGD, FGSM, compared to conventional CNNs. However, without adversarial defense training, it is also relatively vulnerable under these attacks. Note that the gradient of AdderNet is actually artificially designed. The fact that attacking the model with "fake gradient" can also succeed again reveals the weakness of most neural networks.

Further experiments also show that, under black-box transfer attacks from conventional CNN to AdderNet, AdderNet showed relatively high robustness. This again proves the conjecture that the way AdderNet learns features in a somehow different with CNNs.

Besides visualizations and attacks, we also designed a transposed addictive convolution layer that is able to produce an output with larger size than the input. We tried to use this transposed layer to build a GAN, but the result is not very satisfying, meaning that more future works are needed.

For further understand AdderNet, there are still many methods. We can generalise AdderNet to other computer vision tasks, including image segmentation, object detection, etc. We can also explore effective adversarial defense methods for AdderNet. In the forseeable future, if the acceleration methods for AdderNet is well-developed, AdderNet will have the potential to replace CNN in many scenarios, especially when no GPU is unavailable. Therefore, we believe that further researches on this topic will be very meaningful.

References

[1] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization, 2017. 1

[2] Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Addernet: Do we really need multiplications in deep learning?, 2020. 1

[3] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations, 2016. 1

[4] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1, 2016. 1

[5] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009. 2

[6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015. 2, 3, 4

[7] Wenshuo Li, Hanting Chen, Mingqiang Huang, Xinghao Chen, Chunjing Xu, and Yunhe Wang. Winograd algorithm for addernet. *arXiv preprint arXiv:2105.05530*, 2021. 1

[8] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019. 2, 3, 4

[9] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning, 2017. 3

[10] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016. 3, 5

[11] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016. 1

[12] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 2

[13] Dehua Song, Yunhe Wang, Hanting Chen, Chang Xu, Chunjing Xu, and Dacheng Tao. Addersr: Towards energy efficient image super-resolution, 2021. 1

[14] Yunhe Wang, Mingqiang Huang, Kai Han, Hanting Chen, Wei Zhang, Chunjing Xu, and Dacheng Tao. Addernet and its minimalist hardware design for energy-efficient artificial intelligence. *arXiv preprint arXiv:2101.10015*, 2021. 1

[15] Yixing Xu, Chang Xu, Xinghao Chen, Wei Zhang, Chunjing Xu, and Yunhe Wang. Kernel based progressive distillation for adder neural networks. *arXiv preprint arXiv:2009.13044*, 2020. 1

[16] Haoran You, Xiaohan Chen, Yongan Zhang, Chaojian Li, Sicheng Li, Zihao Liu, Zhangyang Wang, and Yingyan Lin. Shiftaddnet: A hardware-inspired deep network. *arXiv preprint arXiv:2010.12785*, 2020. 1

[17] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks, 2013. 2, 4

[18] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, 2018. 1