Scalar Invariant Networks with Zero Bias

Chuqin Geng CHUQIN.GENG@MAIL.MCGILL.CA
Xiaojie Xu XIAOJIE.XU@MAIL.MCGILL.CA
Department of Computer Science, McGill University
Mila - Quebec Artificial Intelligence Insitute

Haolin Ye HAOLIN.YE@MAIL.MCGILL.CA
Department of Computer Science, McGill University

Xujie Si SIX@CS.TORONTO.EDU
Department of Computer Science, University of Toronto

Editors: Sophia Sanborn, Christian Shewmake, Simone Azeglio, Nina Miolane

Abstract

Just like weights, bias terms are learnable parameters in many popular machine learning models, including neural networks. Biases are believed to enhance the representational power of neural networks, enabling them to tackle various tasks in computer vision. Nevertheless, we argue that biases can be disregarded for some image-related tasks such as image classification, by considering the intrinsic distribution of images in the input space and desired model properties from first principles. Our empirical results suggest that zero-bias neural networks can perform comparably to normal networks for practical image classification tasks. Furthermore, we demonstrate that zero-bias neural networks possess a valuable property known as scalar (multiplicative) invariance. This implies that the network's predictions remain unchanged even when the contrast of the input image is altered. We further extend the scalar invariance property to more general cases, thereby attaining robustness within specific convex regions of the input space. We believe dropping bias terms can be considered as a geometric prior when designing neural network architecture for image classification, which shares the spirit of adapting convolutions as the translational invariance prior.

Keywords: Geometric structure; Invariance; Zero Bias; Robustness; Expressive power.

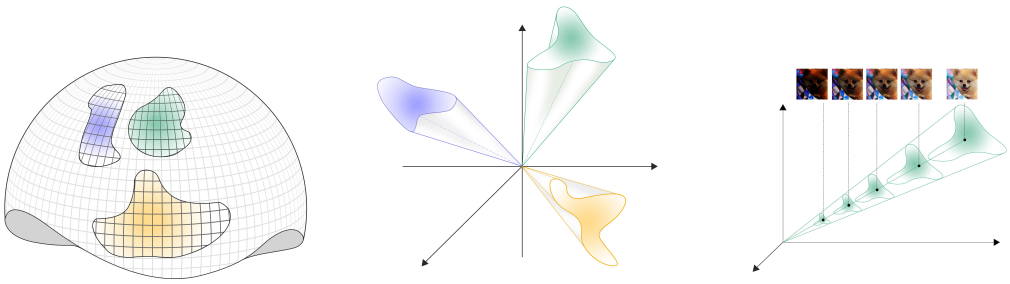
1. Introduction

Using bias terms in neural networks is a common practice. Its theoretical foundation goes back to the invention of artificial neural networks, which are loosely inspired by biological neurons. Biological neurons have some thresholds to determine whether they should "fire" (produce an output that goes to other neurons) (Jain et al., 1996; Yang and Wang, 2020; Hassabis et al., 2017). These thresholds are essentially the same thing as bias terms. From the representation learning perspective, the bias term is widely believed to increase the representational power of neural networks and thus is always needed when designing neural networks to solve a broad array of tasks in computer vision (Wang et al., 2019; Montavon et al., 2018; Alzubaidi et al., 2021).

In this work, we challenge the commonly-held beliefs of the necessity of including bias terms in neural networks to solve image classification tasks. Our geometric observations suggest the intrinsic distribution of images should incorporate *directionality*, as suggested in

© 2023 C. Geng, X. Xu, H. Ye & X. Si.

Figure 1. With this property holding, bias terms should not affect models' representational power and performance, even for large modern CNN models such as ResNets (He et al., 2015b). Indeed, several recent works like SphereFace (Liu et al., 2017a) and SphereNet (Liu et al., 2017b) achieve strong performance in real-world tasks by ignoring the bias term and designing angular-inspired losses. Moreover, Hesse et al. (2021) reports that removing bias terms only has a minor impact on predictive accuracy. Our thorough experimental results also support this argument. In addition, we show that neural networks will possess



(a) Image classes can be thought of as manifolds in n-dimensional input space.

(b) Image classes can be conceptualized as corn-shaped if considering the varying contrast.

(c) An image specifes a direction in input space, and copies of that image with varying contrast lie along the same direction.

Figure 1: The directionality manifests in the intrinsic distribution of images.

an intriguing property - *scalar (multiplication) invariance* after dropping bias terms. We then extend scalar invariance to CNNs as well as ResNets. This property allows zero-bias networks to perfectly generalize to inputs with different levels of contrast without any data augmentation, which normal neural networks (with biases) usually fail to do so. Based on the scalar invariance property, we further derive more general robustness guarantees that could verify even certain convex regions of input space. In contrast, normal neural networks are highly combinatorial in nature, making such guarantees hardly exist.

We summarize our contributions as follows: (1) We show that the basic building blocks of neural networks are scalar multiplication associative if the bias is ignored. This, in turn, assures the scalar invariant property of convolutional neural networks. By adapting batch normalization-free methods, we can extend scalar invariance to ResNets. We also conduct experiments on a few popular image classification benchmarks to validate the scalar invariant property; (2) Based on the scalar invariant property, we propose two additional robustness properties that verify inputs along certain lines (interpolations) and convex regions of the input space. Empirical validation of the interpolation robustness guarantee is done using image examples from benchmarks such as MNIST and CIFAR-10; (3) Our geometric observations suggest the intrinsic distribution of images should incorporate *directionality*. Under this property, scalar invariant neural networks should have the same representational power as normal neural networks, thus delivering comparable performances.

2. Scalar Invariant Neural Networks

2.1. Preliminary

A neural network consists of an input layer, hidden layers, and an output layer. For convolutional neural networks, some of the hidden layers are called convolution layers which perform convolution operations on their input tensors with convolution kernels. The outputted tensors are passed to an activation function, commonly *ReLU*, before downsampling through pooling layers. After that, the input tensor is flattened out so that a fully connected network can process it and calculate the final prediction. For classification tasks, the final prediction is represented by a probability distribution over all classes using some activation functions such as *Softmax*. To further investigate the scalar invariant property, we formally denote the input tensor as X and a convolutional neural network as $\mathcal{N}$. Then $\mathcal{N}$ is composed of convolutional layers $\mathcal{F}_i$, pooling layers $\mathcal{P}_i$, and fully connected layers $\mathcal{L}_j$, where $i, j \in \mathbb{N}$. And we denote the final activation function as $\mathcal{A}$ and *ReLU* as $\mathcal{R}$. We think of layers and activation functions as transformations on the input X, then the output of the network before the final activation function $\mathcal{A}$ is represented by:

$$\mathcal{O}(X) = \underbrace{\mathcal{L}_j \circ \mathcal{R} \circ ... \circ \mathcal{R} \circ \mathcal{L}_1}_{\text{j Linear layers}} \circ \underbrace{\mathcal{P}_i \circ \mathcal{R} \circ \mathcal{F}_i ... \circ \mathcal{P}_1 \circ \mathcal{R} \circ \mathcal{F}_1}_{\text{i Convolutional layers}} \circ X \tag{1}$$

And the final prediction class is determined by the one with the highest probability over all classes $\mathcal{C}$, that is:

$$\mathcal{N}(X) = \underset{c \in \mathcal{C}}{\operatorname{argmax}} \{\mathcal{A} \circ \mathcal{O}(X)\} \tag{2}$$

2.2. Scalar Associative Transformations

We consider the operation inside a convolution layer $\mathcal{F}$ with a kernel $\mathcal{K}$, it is easy to show the associative property with scalar multiplication hold for convolution operations. More formally, let s be a **positive** scalar s.t. $s \in \mathbb{R}^+$, then we have:

$$\mathcal{F} \circ (sX) = \sum_m \sum_n sX(i+m, j+n)\mathcal{K}(m,n) = s \sum_m \sum_n X(i+m, j+n)\mathcal{K}(m,n) = s(\mathcal{F} \circ X) \tag{3}$$

In addition, the above property also holds for pooling layers $\mathcal{P}$, including max pooling and average pooling. Since both the max and average operation should preserve the scalar multiplication. The same argument also applies to the *ReLU* function. So we have:

$$\mathcal{P} \circ (sX) = s(\mathcal{P} \circ X) \ \textit{and} \ \mathcal{R} \circ (sX) = s(\mathcal{R} \circ X) \tag{4}$$

Finally, passing the input X to a fully connected layer $\mathcal{L}$ can be thought of as applying a linear transformation $(\mathcal{W}, \mathcal{B})$ on X. If we set the bias term $\mathcal{B}$ to $\mathbf{0}$, we will have the scalar associative property. That is:

$$\mathcal{L} \circ (sX) = (sX)\mathcal{W}^T = sX\mathcal{W}^T = s(\mathcal{L} \circ X) \tag{5}$$

Note our proofs also use the commutative property which generally holds for matrix and vector multiplications with a scalar. Put together, by setting biases to zeros, we have the scalar (multiplication) associative property holds for the output function, i.e., $(\mathcal{O}(sX) = s\mathcal{O}(X))$.

2.3. Scalar Invariant Convolutional Neural Networks

Now we consider how to calculate the final prediction of the network $\mathcal{N}$. For classification tasks, the last activation function $\mathcal{A}$ is usually *Softmax*. If we multiply the input X with a scalar s ($s \in \mathbb{R}^+$) and pass the product to *Softmax*, it is equivalent to changing the temperature of the distribution. Note that the rank of candidate classes remains the same despite the change in the shape of the distribution. In other words, the predicted class by the network $\mathcal{N}$ is scalar (multiplication) invariant:

$$\operatorname*{argmax}_c \frac{e^{s\mathcal{O}(X)_c}}{\sum_{c\in\mathcal{C}} e^{s\mathcal{O}(X)_c}} = \operatorname*{argmax}_c \frac{e^{\mathcal{O}(X)_c}}{\sum_{c\in\mathcal{C}} e^{\mathcal{O}(X)_c}} \tag{6}$$

Put together with the scalar associative property of the output function $\mathcal{O}(\cdot)$, we have a scalar invariant neural network:

$$\mathcal{N}(sX) = \operatorname*{argmax}_c \{\mathcal{A} \circ \mathcal{O}(sX)\} = \operatorname*{argmax}_c \{\mathcal{A} \circ \mathcal{O}(X)\} = \mathcal{N}(X) \tag{7}$$

The concept of scalar invariant neural networks generalizes beyond just convolutional neural networks. In fact, as long as hidden layers perform scalar associative (and commutative) transformations and the last activation function preserves the highest probable candidate under scalar multiplication, the neural network will be scalar invariant. Since an image input X represents a direction in the input space and we have proved that zero-bias neural networks could yield the same prediction along that direction, we could restate this property as directional robustness property.

Lemma 1 (Directional robustness property) *For any input X to a zero-bias neural network $\mathcal{N}$, the prediction remains the same when X is multiplied by any positive scalar s. Formally, we have $\mathcal{N}(sX) = \mathcal{N}(X) \quad \forall s \in \mathbb{R}^+$.*

2.4. Scalar Invariant ResNet

We briefly discussed the most simple architecture of convolutional neural networks in the previous section. However, in addition to those basic layers we mention before, modern powerful CNNs also employ extra layers and techniques to address over-fitting and gradient exploding/vanishing issues. For example, ResNet (He et al., 2015b) adopts *Dropout* (Srivastava et al., 2014), *Additive Skip Connection* (He et al., 2015b) and *Batch Normalization* (Ioffe and Szegedy, 2015) which contributes enormously to its success. First, as dropout layers are disabled during the inference phase, it has no impact on the scalar invariant property. Second, it is trivial to show skip connection is also scalar multiplication associative if the corresponding residual branch $\mathcal{G}$ is also scalar multiplication associative.

$$sX + \mathcal{G}(sX) = s(X + \mathcal{G}(X)) \quad \forall s \in \mathbb{R}^+ \tag{8}$$

Lastly, we consider Batch Normalization, which is performed through a normalization transformation that fixes the means and variances of inputs to each layer. Let us use $X_{\mathcal{B}}$ to denote

a mini-batch of the entire training set. Then we have the batch normalization transformation as follows:

$$\mathcal{BN}(X_{\mathcal{B}}) = \gamma \hat{X}_{\mathcal{B}} + \beta \tag{9}$$

where γ and β are learnable parameters, and $\hat{X}_{\mathcal{B}}$ is the normalized input, represented by $\hat{X}_{\mathcal{B}} = \frac{X_{\mathcal{B}} - \mu_{\mathcal{B}}}{\sqrt{(\sigma_{\mathcal{B}})^2 + \epsilon}}$, ϵ is an arbitrarily small constant. Clearly, we observe that the scalar associative/invariant property doesn't hold for the normalization step, because:

$$\gamma(sX) + \beta = \gamma \frac{(sX) - \mu_{\mathcal{B}}}{\sqrt{(\sigma_{\mathcal{B}})^2 + \epsilon}} + \beta \neq s(\gamma X + \beta) \tag{10}$$

Thus, in order to achieve scalar invariance, we can adopt two approaches. Firstly, for small neural networks that do not have severe gradient explosion/vanishing issues, we can drop $\mathcal{BN}$ layers. Secondly, for larger neural networks, we can consider some alternatives to batch normalization. There exists a line of work on exploring efficient residual learning without normalization such as Instance Normalization (Ulyanov et al., 2016), Fixup (Zhang et al., 2019), $\mathcal{X}$-DNNs (Hesse et al., 2021), and NFNets (Brock et al., 2021a,b). The majority of these approaches can be easily adapted to achieve scalar invariance, further information can be found in Appendix B.

Table 1: As expected, zero-bias neural networks achieve perfect scalar invariance on testing accuracies, while normal neural networks are generally not robust against decreasing the contrast of the input image. Results are replicated thrice and averaged to reduce stochasticity effects, with all variances being below 0.5.

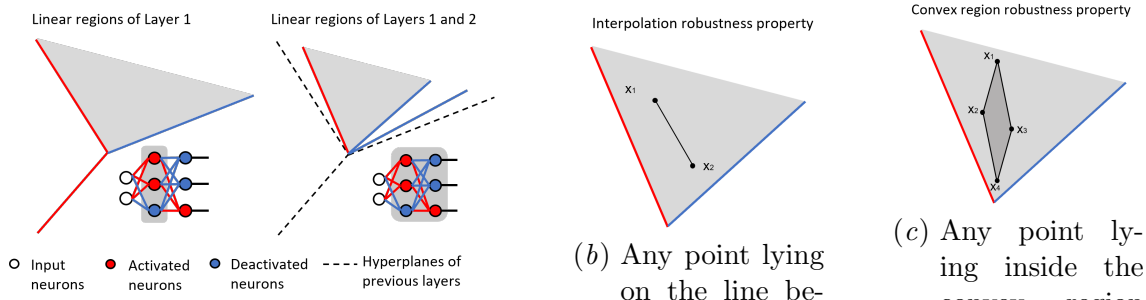
			Scalar multiplier										
			1	0.25	0.15	0.125	0.1	0.075	0.05	0.025	0.01	0.001	0.0001
MNIST	FCN	w/ bias	88.12	87.07	84.46	82.57	79.52	74.76	65.82	42.84	16.34	10.28	10.28
		w/o bias	**88.27**	**88.27**	**88.27**	**88.27**	**88.27**	**88.27**	**88.27**	**88.27**	**88.27**	**88.27**	**88.27**
Fashion-MNIST	CNN	w/ bias	**89.10**	67.10	40.12	32.52	24.16	17.91	12.46	10.12	10.00	10.00	10.00
		w/o bias	89.02	**89.02**	**89.02**	**89.02**	**89.02**	**89.02**	**89.02**	**89.02**	**89.02**	**89.02**	**89.02**
CIFAR-100	ResNet18	w/ bias	**67.62**	19.86	8.20	6.11	4.16	2.58	1.69	1.06	1.01	1.01	1.01
		w/o bias	67.33	**67.33**	**67.33**	**67.33**	**67.33**	**67.33**	**67.33**	**67.33**	**67.33**	**67.33**	**67.33**
ImageNet (Deng et al., 2009)	ResNet50	w/ bias	**75.37**	66.72	57.84	53.62	47.27	37.61	21.81	3.39	0.21	0.10	0.10
		w/o bias	73.82	**73.82**	**73.82**	**73.82**	**73.82**	**73.82**	**73.82**	**73.82**	**73.82**	**73.82**	**73.82**

2.5. Scalar Invariance Evaluation

In this section, we conduct a series of experiments to verify the scalar invariance property of zero-bias neural networks and their normally trained counterparts. We train both types of neural networks using the same configuration, except for the option of using bias, on several popular image classification benchmarks. More training details can be found in Appendix C. We further demonstrate the effect of scalar invariance by evaluating their accuracy on test sets multiplied by different scalars, ranging from 1 to 0.0001. The results, which are presented in Table 1, suggest that zero-bias networks and normal networks achieve similar accuracies when the scalar is set to 1. However, when the contrast/scalar multiplier of the input image decreases, normal networks show a lack of robustness as their accuracy declines at varying rates. In contrast, zero-bias networks achieve scalar invariance as expected, and

their performance remains unchanged regardless of the varying contrast of input images. We also train both types of models using augmented training sets that involve multiplication of the scalars used in test evaluation. We find that with-bias models trained on augmented data still perform poorly when the scalar multiplier is extremely small, such as 0.001 and 0.0001. For larger scalar ranges from 0.25 to 0.01, with-bias models are merely comparable to zero-bias models. These results demonstrate a significant advantage of zero-bias networks in terms of data efficiency.

3. Interesting Robustness Properties



(a) Core-shape unbounded regions of zero-bias neural networks are formed by hyper-planes that pass through the origin.

(b) Any point lying on the line between x_1 and x_2 should yield the same prediction as x_1 and x_2.

(c) Any point lying inside the convex region formed by inputs x_i should yield the same prediction as x_i.

Figure 2: Zero-bias neural networks exhibit unique robustness properties when inputs share the same neural activation pattern and are predicted identically by the neural network.

Despite achieving remarkable success in a wide range of tasks, neural networks have been proven not robust under even small perturbations to the input (Carlini et al., 2019; Akhtar and Mian, 2018), which accelerates the study of neural network verification and attacks. We find that zero-bias networks exhibit some interesting robustness guarantees that are rarely identified in common neural networks.

Given that these guarantees of robustness are closely tied to specific regions within the input space, it is pertinent to explore how zero-bias neural networks divide the geometry of the input space. When the bias terms are eliminated, the hyperplanes defined by each neuron will originate from the origin. When these hyperplane arrange together, they create multiple core-shaped unbounded regions that differ from the typical convex regions formed by normal neural networks, as illustrated in Figure 2(a). To better illustrate the interesting robustness properties of zero-bias networks, we first introduce the notion of neural activation patterns (Geng et al., 2023).

Definition 2 (Neural activation pattern) *A* Neural Activation Pattern (NAP) *of a neural network $\mathcal{N}$ is a tuple $\mathcal{P}_x := (A, D)$, where A and D are collections of all activated and deactivated neurons respectively when passing x through $\mathcal{N}$.*

Theorem 3 (Interpolation robustness property) *For any two inputs X_1 and X_2 that have the same prediction and neural activation pattern by network $\mathcal{N}$, i.e., $\mathcal{N}(X_1) = \mathcal{N}(X_2)$ and $\mathcal{P}_{X_1} = \mathcal{P}_{X_2}$, their linear interpolation also yield the same prediction, that is, $\mathcal{N}(\lambda X_1 + (1-\lambda)X_2) = \mathcal{N}(X_1) = \mathcal{N}(X_2)$, where $\lambda \in [0, 1]$.*

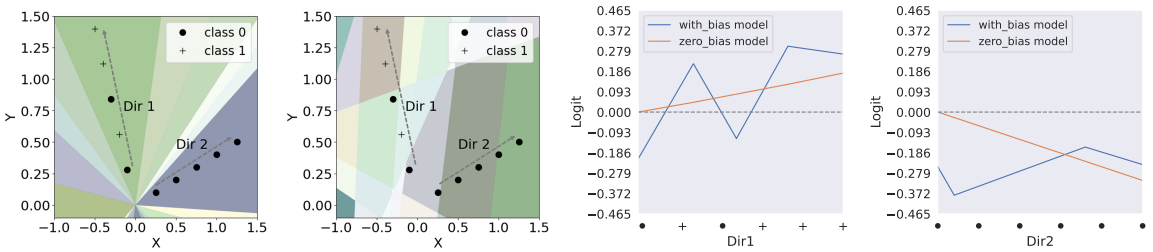
Assuming that two points share the same prediction and neural activation pattern, it can be proven that their interpolation will also share the same prediction and neural activation pattern. Please refer to Appendix A for detailed proof. What's even more interesting is that this property can be extended to the multiple inputs setting, where a convex region can provide robustness assurance.

Theorem 4 (Convex region robustness property) *Let $\{X_i \mid i \in \{1, 2, \ldots, n\}\}$ be a collection of inputs that have the same prediction and neural activation pattern by network $\mathcal{N}$, we denote the convex polygon formed by vertices X_i as $\mathcal{M}$. Then, for any point m that lies inside the polygon $\mathcal{M}$, m also yield the same prediction as X_i, that is, $\mathcal{N}(m) = \mathcal{N}(X_i) \quad \forall m \in \mathcal{M}\ \forall i \in \{1, 2, ..., n\}$.*

As m can always be represented by some linear combination of vertices $\{X_i\}$, the convex region robustness property holds as the direct result of **Theorem** 3. In contrast, such guarantees hardly exist on normal neural networks due to their highly combinatorial nature. Furthermore, recent research has shown that ignoring bias can enhance the robustness of models, as demonstrated in Diffenderfer and Kailkhura (2021); Diffenderfer et al. (2021).

To test the interpolation robustness property, we conducted experiments using visual examples sourced from MNIST and CIFAR-10. Following neural network training, we search for image pairs that shared the same prediction and neural activation pattern. We then interpolate 1000 images between each pair and confirm that each interpolation yields the same prediction, as expected. Please refer to Appendix D for some examples of our findings.

However, it is important to note that such robustness guarantees are rarely identified in larger and more accurate neural networks. For example, with the small neural networks used in our experiments comprising only 30 neurons and achieving accuracies of 32.27% and 29.6% on MNIST and CIFAR-10, respectively, we can easily identify many qualified pairs. In larger networks with accuracies of around 80%, we still find a few qualified pairs. However, in even larger networks with accuracies of over 90%, we are unable to find any examples of the interpolation robustness property. This is due to the fact that as the number of neurons grows, the input space becomes more scattered, reducing the likelihood of two or more inputs sharing the same neural activation pattern. While this seems like a new *No Free Lunch Theorem* in terms of the trade-off between interpolation robustness and accuracy, we believe there are methods to improve model accuracy while still maintaining these robustness guarantees. For example, we could design new training objectives to control the diminishing margin between hyperplanes of networks. We leave this for future work.



(a) (b) (c) (d)

Figure 3: Geometric insights into the expressive power of zero-bias networks and normal networks. (a) *Dir* 1 and *Dir* 2 stay within unbounded regions of the zero-bias network. (b) *Dir* 1 and *Dir* 2 traverse multiple convex regions of the with-bias network. (c) Normal networks can fit data in *Dir* 1 whereas zero-bias networks fail to do so. (d) Both normal and zero-bias networks can fit data in *Dir* 2.

4. Geometric Insights on Expressiveness

It is a widely held belief that eliminating bias from neural networks can diminish their representational power, ultimately affecting the accuracy of models. For instance, Xu et al. (2021) shows that neural networks linearize outside of their training regime once omitting the biases. To this end, we provide geometric insights to show that zero-bias networks are comparable with normal networks in expressive capabilities for image classification tasks.

Since a neural network can be thought of as a piece-wise (linear) function defined over many convex polytopes (Hanin and Rolnick, 2019a,b), we plot linear regions of a simple 3-layer neural network and its zero-bias counterpart on a simple 2D input space to study their representational power in Figure 3. Our aim is to illustrate how these networks perform on two simple binary classification tasks characterized by *Dir (Direction)* 1 and *Dir* 2. In *Dir* 1, points along the same direction are not labelled identically, whereas points along *Dir* 2 are assigned to the same class, i.e., satisfying *directionality*. Note that in this study, we say a model can fit a specific point if its prediction logit function (before *Sigmoid*) is negative for $\bullet$ and positive for $+$. While zero-bias networks have a more limited expressive capacity compared to normal networks, being restricted to linear functions originating from the origin, they can effortlessly fit *Dir* 2, as shown in Figure 3(d). This is due to the fact that all points in the predicted logit fall below 0. However, in the absence of directionality - *Dir* 1, shown in Figure 3(c), the points in the predicted logit scatter across the 0 line (as they belong to different classes). A linear function starting from the origin could never fit this case. On the other hand, normal networks, with their highly expressive piece-wise functions, can fit both *Dir* 1 and *Dir* 2.

In conclusion, our observations on the expressive power of the two types of neural networks are consistent with our experimental findings in Section 2.5. Moreover, we propose that directionality can serve as a powerful geometric prior in image classification, akin to the translational invariance prior employed in CNNs.

5. Related work

5.1. Invariance in Neural Networks

Studying invariance in machine learning as well as neural networks has attracted much attention as real-world data such as images often exhibit rich invariant structures. Incorporating such invariance properties as prior knowledge (inductive bias) could expand the expressive power of the network without much increase in the number of parameters, which usually leads to better performance. For instance, Convolutional Neural Networks have a stronger geometric prior - translation invariance (Bronstein et al., 2021; Bouvrie, 2006). In addition, Group equivariant Convolutional Neural Networks (G-CNNs) adapt group convolution layers to achieve great results on images generated by translations, reflections, and rotations (Cohen and Welling, 2016). Similar work also focuses on studying the invariance of neural network's outputs under group actions on its inputs (Kondor and Trivedi, 2018; Lyle et al., 2020; Bloem-Reddy and Teh, 2020).

Given the scale invariant nature of images, there is also a line of work studies how to improve the consistency of models' prediction on varying scale images (Xu et al., 2014; Ghosh and Gupta, 2019; Perantonis and Lisboa, 1992; Zhang et al., 2017b). However, the most related invariance to our work is illumination invariance which has a great impact on many real-world applications. For example, Ramaiah et al. (2015) uses convolutional neural networks for face recognition under non-uniform illumination. Maddern et al. (2014) studies illumination invariant transform to improve visual localization, mapping, and scene classification for autonomous road vehicles. Despite absolute invariance being considered hard to achieve and most works usually failing to guarantee it, our work shows that *absolute invariance* under scalar multiplication can be achieved with zero-bias neural networks.

5.2. Zero-bias Neural Networks

Although zero-bias neural networks do not appear as much as normal neural networks in the machine-learning literature due to potential reductions in models' expressive capability, they have been used in some real-world applications such as facial expression recognition (Khorrami et al., 2015), identification of Internet-of-Things devices (Liu et al., 2021b), RF signal surveillance (Liu et al., 2021a), and anomaly data detection (Zhang et al., 2021). There are several reasons for choosing zero-bias neural networks over normal neural networks: (1) Their incremental learning fashion and better decision fairness; (2) Better interpretability without losing accuracy, which challenges the common first impression of the weaker expressive capability of zero-bias models; (3) More reliable and robust performance. Although these works achieve some success with zero-bias neural networks, none of them dive deeper to analyze these advantages formally. Our work explores zero-bias from an invariant perspective for the first time, to our best knowledge, identifying scalar multiplication invariance in zero-bias models, proving some rigorous robust guarantees, and explaining their comparable accuracy based on geometric sights of image distribution.

6. Conclusion

In this paper, we study how neural networks behave after removing bias terms, with a focus on image classification. We prove that, by simply dropping bias terms, the prediction of

neural networks achieves absolute invariance under varying contrast of the input image. Moreover, derived from the scalar invariance property, we show that zero-bias networks are robust on certain lines and convex regions of the input space. Although it is commonly believed that bias improves models' expressive capability and thus is always needed, we show that it can be completely ignored for image classification tasks if we consider directionality, an important property of the intrinsic distribution of images. Finally, we believe dropping bias terms shares the spirit of adapting convolutions as a strong prior in designing neural network architecture in computer vision. We consider this work as a preliminary step towards comprehending zero-bias networks and intend to further explore their robustness, fairness, and generalization ability in our future research.

References

Naveed Akhtar and Ajmal S. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-dujaili, Ye Duan, Omran Al-Shamma, Jesus Santamaría, Mohammed Abdulraheem Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, 8, 2021.

Benjamin Bloem-Reddy and Yee Whye Teh. Probabilistic symmetries and invariant neural networks. *J. Mach. Learn. Res.*, 21:90:1–90:61, 2020.

Jake Bouvrie. Notes on convolutional neural networks. *CoRR*, 2006.

Andrew Brock, Soham De, and Samuel L. Smith. Characterizing signal propagation to close the performance gap in unnormalized resnets. In *ICLR*. OpenReview.net, 2021a.

Andy Brock, Soham De, Samuel L. Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 1059–1071. PMLR, 2021b.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, abs/2104.13478, 2021.

Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian J. Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *CoRR*, abs/1902.06705, 2019. URL http://arxiv.org/abs/1902.06705.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/cohenc16.html.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

James Diffenderfer and Bhavya Kailkhura. Multi-prize lottery ticket hypothesis: Finding accurate binary neural networks by pruning A randomly weighted network. *CoRR*, abs/2103.09377, 2021. URL https://arxiv.org/abs/2103.09377.

James Diffenderfer, Brian R. Bartoldson, Shreya Chaganti, Jize Zhang, and Bhavya Kailkhura. A winning hand: Compressing deep networks can improve out-of-distribution robustness. *CoRR*, abs/2106.09129, 2021. URL https://arxiv.org/abs/2106.09129.

Chuqin Geng, Nham Le, Xiaojie Xu, Zhaoyue Wang, Arie Gurfinkel, and Xujie Si. Towards reliable neural specifications. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 11196–11212. PMLR, 2023. URL https://proceedings.mlr.press/v202/geng23a.html.

Rohan Ghosh and Anupam K. Gupta. Scale steerable filters for locally scale-invariant convolutional neural networks. *CoRR*, abs/1906.03861, 2019.

Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 2596–2604. PMLR, 2019a.

Boris Hanin and David Rolnick. Deep relu networks have surprisingly few activation patterns. In *NeurIPS*, pages 359–368, 2019b.

Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2017.06.011. URL https://www.sciencedirect.com/science/article/pii/S0896627317305093.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034. IEEE Computer Society, 2015a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015b. URL https://arxiv.org/abs/1512.03385.

Robin Hesse, Simone Schaub-Meyer, and Stefan Roth. Fast axiomatic attribution for neural networks. *CoRR*, abs/2111.07668, 2021. URL https://arxiv.org/abs/2111.07668.

Jeremy Howard. Imagenette: A smaller subset of 10 easily classified classes from imagenet, March 2019. URL https://github.com/fastai/imagenette.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

A.K. Jain, Jianchang Mao, and K.M. Mohiuddin. Artificial neural networks: a tutorial. *Computer*, 29(3):31–44, 1996. doi: 10.1109/2.485891.

Pooya Khorrami, Thomas Paine, and Thomas Huang. Do deep neural networks learn facial action units when doing expression recognition? In *Proceedings of the IEEE international conference on computer vision workshops*, pages 19–27, 2015.

Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 2752–2760. PMLR, 2018.

Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, 2017a.

Weiyang Liu, Yan-Ming Zhang, Xingguo Li, Zhiding Yu, Bo Dai, Tuo Zhao, and Le Song. Deep hyperspherical learning. In *NIPS*, 2017b.

Yongxin Liu, Yingjie Chen, Jian Wang, Shuteng Niu, Dahai Liu, and Houbing Song. Zero-bias deep neural network for quickest rf signal surveillance. In *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 1–8, 2021a. doi: 10.1109/IPCCC51483.2021.9679426.

Yongxin Liu, Jian Wang, Jianqiang Li, Houbing Song, Thomas Yang, Shuteng Niu, and Zhong Ming. Zero-bias deep learning for accurate identification of internet-of-things (iot) devices. *IEEE Internet of Things Journal*, 8(4):2627–2634, 2021b. doi: 10.1109/JIOT.2020.3018677.

Clare Lyle, Mark van der Wilk, Marta Kwiatkowska, Yarin Gal, and Benjamin Bloem-Reddy. On the benefits of invariance in neural networks. *CoRR*, abs/2005.00178, 2020.

Will Maddern, Alex Stewart, Colin McManus, Ben Upcroft, Winston Churchill, and Paul Newman. Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles. In *Proceedings of the Visual Place Recognition in Changing Environments Workshop, IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China*, volume 2, page 5, 2014.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018. ISSN 1051-2004. doi: https://doi.org/10.1016/j.dsp.2017.10.011. URL https://www.sciencedirect.com/science/article/pii/S1051200417302385.

S.J. Perantonis and P.J.G. Lisboa. Translation, rotation, and scale invariant pattern recognition by high-order neural networks and moment classifiers. *IEEE Transactions on Neural Networks*, 3(2):241–251, 1992. doi: 10.1109/72.125865.

N. Pattabhi Ramaiah, Earnest Paul Ijjina, and C. Krishna Mohan. Illumination invariant face recognition using convolutional neural networks. In *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, pages 1–4, 2015. doi: 10.1109/SPICES.2015.7091490.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016. URL http://arxiv.org/abs/1607.08022.

Shengjie Wang, Tianyi Zhou, and Jeff Bilmes. Bias also matters: Bias attribution for deep neural network explanation. In *International Conference on Machine Learning*, pages 6659–6667. PMLR, 2019.

Keyulu Xu, Mozhi Zhang, Jingling Li, Simon Shaolei Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=UH-cmocLJC.

Yichong Xu, Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, and Zheng Zhang. Scale-invariant convolutional neural networks. *CoRR*, abs/1411.6369, 2014. URL http://arxiv.org/abs/1411.6369.

Guangyu Robert Yang and Xiao-Jing Wang. Artificial neural networks for neuroscientists: A primer. *Neuron*, 107(6):1048–1070, 2020. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2020.09.005. URL https://www.sciencedirect.com/science/article/pii/S0896627320307054.

Bo Zhang, Qiang Zhang, Yong Xin Liu, and Ou Ye. Anomaly data detection for ads-b based on zero-bias inception network. In *2021 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pages 1–6, 2021. doi: 10.1109/ICSPCC52875.2021.9564587.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017a. URL http://arxiv.org/abs/1710.09412.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *ICLR (Poster)*. OpenReview.net, 2019.

Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z Li. S3fd: Single shot scale-invariant face detector. In *Proceedings of the IEEE international conference on computer vision*, pages 192–201, 2017b.

Appendix A. Proof of Interpolation Robustness Property

Theorem 3(Interpolation robustness property) *For any two inputs X_1 and X_2 that have the same prediction and neural activation pattern by network $\mathcal{N}$, i.e., $\mathcal{N}(X_1) = \mathcal{N}(X_2)$ and $\mathcal{P}_{X_1} = \mathcal{P}_{X_2}$, their linear interpolation also yield the same prediction, that is, $\mathcal{N}(\lambda X_1 + (1-\lambda)X_2) = \mathcal{N}(X_1) = \mathcal{N}(X_2)$, where $\lambda \in [0, 1]$.*

Proof We show the interpolation robustness property holds for fully connected neural networks without bias. For more complicated neural networks such as CNN, the property also holds as long as all transformations before the output layer are scalar associative (**Lemma** 1). Consider a FCN $\mathcal{N}$ composed of J number of fully connected layers $\mathcal{L}_j$ and some *ReLU* layers $\mathcal{R}$. We think of layers and activation functions as transformations on the input X, then the output of the network $\mathcal{O}(\lambda X_1 + (1-\lambda)X_2)$ before the *softamx* function is represented by:

$$\mathcal{O}(\lambda X_1 + (1-\lambda)X_2) = \mathcal{L}_J \circ \mathcal{R} \circ ... \circ \mathcal{R} \circ \mathcal{L}_1 \circ (\lambda X_1 + (1-\lambda)X_2) \tag{11}$$

For any fully connected layer $\mathcal{L}_j$, we have:

$$\mathcal{L}_j \circ (\lambda X_1 + (1-\lambda)X_2) = (\lambda X_1 + (1-\lambda)X_2)\mathcal{W}_j^T = \lambda X W_j^T + (1-\lambda)X_2\mathcal{W}_j^T = \lambda \mathcal{L}_j \circ X_1 + (1-\lambda)\mathcal{L}_j \circ X_2 \tag{12}$$

On the other hand, we have X and Y falling into the same neural activation pattern. Since the linear region corresponding to the neural activation pattern is convex, the interpolation of X_1 and X_2, $\lambda X_1 + (1-\lambda)X_2$, also lies in the same neural activation pattern. Furthermore, we have:

$$\begin{gathered}
\mathcal{R} \circ \mathcal{L}_1 \circ (\lambda X_1 + (1-\lambda)X_2) = \lambda \mathcal{R} \circ \mathcal{L}_1 \circ X_1 + (1-\lambda)\mathcal{R} \circ \mathcal{L}_1 \circ X_2 \\
\mathcal{R} \circ \mathcal{L}_2 \circ (\lambda \mathcal{R} \circ \mathcal{L}_1 \circ X_1 + (1-\lambda)\mathcal{R} \circ \mathcal{L}_1 \circ X_2) = \lambda \mathcal{R} \circ \mathcal{L}_2 \circ \mathcal{R} \circ \mathcal{L}_1 \circ X + (1-\lambda)\mathcal{R} \circ \mathcal{L}_2 \circ \mathcal{R} \circ \mathcal{L}_1 \circ X_2 \\
... \\
\mathcal{O}(\lambda X_1 + (1-\lambda)X_2) = \lambda \mathcal{O}(X_1) + (1-\lambda)\mathcal{O}(X_2), \text{by } \textbf{Lemma}1
\end{gathered} \tag{13}$$

Given that $\mathcal{N}(X_1) = \mathcal{N}(X_2)$, the index/class of the highest logit of $\mathcal{O}(X_1)$ and $\mathcal{O}(X_2)$ must be the same, that is:

$$\operatorname*{argmax}_c \mathcal{O}(X_1)_c = \operatorname*{argmax}_c \mathcal{O}(X_2)_c \tag{14}$$

Since multiplying a positive scalar to the operand won't change the output of theargmax operator, we have:

$$\operatorname*{argmax}_c \lambda \mathcal{O}(X_1)_c = \operatorname*{argmax}_c (1-\lambda)\mathcal{O}(X_2)_c = \operatorname*{argmax}_c \mathcal{O}(X_1)_c = \operatorname*{argmax}_c \mathcal{O}(X_2)_c \tag{15}$$

Note that the index/class of the highest logit of $\lambda\mathcal{O}(X_1)_c$ and $(1-\lambda)\mathcal{O}(X_2)_c$ are the same, the index/class of the highest logit of their addition is also the same as $\lambda\mathcal{O}(X_1)_c$ and $(1-\lambda)\mathcal{O}(X_2)_c$. Then it follows that:

$$\operatorname*{argmax}_c \mathcal{O}(\lambda X_1 + (1-\lambda)X_2)_c = \operatorname*{argmax}_c \mathcal{O}(X_1)_c = \operatorname*{argmax}_c \mathcal{O}(X_2)_c \tag{16}$$

Since the *softmax* function will preserve the ranking of logits, we have:

$$\underset{c}{\operatorname{argmax}} \frac{e^{\mathcal{O}(\lambda X_1+(1-\lambda)X_2)_c}}{\sum_{c\in\mathcal{C}} e^{\mathcal{O}(\lambda X_1+(1-\lambda)X_2)_c}} = \underset{c}{\operatorname{argmax}} \frac{e^{\mathcal{O}(X_1)_c}}{\sum_{c\in\mathcal{C}} e^{\mathcal{O}(X_1)_c}} = \underset{c}{\operatorname{argmax}} \frac{e^{\mathcal{O}(X_2)_c}}{\sum_{c\in\mathcal{C}} e^{\mathcal{O}(X_2)_c}} \tag{17}$$

Finally, this can be restated as:

$$\mathcal{N}(\lambda X_1 + (1-\lambda)X_2) = \mathcal{N}(X_1) = \mathcal{N}(X_2) \tag{18}$$

■

Appendix B. Batch Normalization Free Methods

Fixup enables training deep residual networks with comparable performance in terms of convergence, generalization, etc, without normalization. More specifically, this method rescales the standard initialization of residual branches by taking the network architecture into account. The key steps of Fixup initialization are described as follows:

1. Initialize the last layer of each residual branch and the classification layer to 0.
2. Initialize other layers using a standard method (He et al., 2015a), and scale only the weight layers inside residual branches by $L^{-\frac{1}{2m-2}}$, where L and m are the numbers of residual blocks and layers inside a residual branch respectively.
3. Add a scalar multiplier before each convolution, linear, and element-wise activation layer in each residual branch, the multiplier is initialized at 1 [1].

It is obvious that the above initialization steps perform some transformations on the weights of neural networks instead of the input, and the scalar multiplier is scalar associative which ensures the trained ResNet is scalar invariant.

NFNet aims to overcome the same challenge of developing ResNet variants without normalization layers yet is comparable to batch-normalized ResNets in many aspects. The effect of standard batch normalization operation within each residual block can be summarized as: 1) downscales the input by a factor proportional to its standard deviation; 2) increases the variance of the input signal by an approximately constant factor. By mimicking the effect of batch normalization, the residual blocks can be written in the form of $X_{l+1} = X_l + \alpha\mathcal{G}_l(X_l/\beta_l)$, where X_l denotes the input to the lth residual block and $\mathcal{G}_l(\cdot)$ denotes the lth residual branch. Moreover, the network should be designed such that:

- $\mathcal{G}_l(\cdot)$ is parameterized to be able to preserve variance at initialization, i.e., $Var(\mathcal{G}_l(z)) = Var(z)$ for all l.
- β_l is a fixed scalar, set it to be $\sqrt{Var(X_l)}$, the expected empirical standard deviation of X_l at initialization.
- α is a hyperparameter that controls the growth rate of variance between blocks.

Since both α and β are fixed scalar during the inference phase. The modified residual blocks are scalar associative since $sX_l + \alpha\mathcal{G}_l(sX_l/\beta_l) = s(X_l + \alpha\mathcal{G}_l(X_l/\beta_l))$. We conclude the NFNet method also ensures scalar invariance.

1. We intentionally ignore the scalar bias (initialized at 0) presented in the original paper to ensure scalar invariance.

Appendix C. Training Details

Our ResNet50 model, as shown in Table 1, is trained for 100 epochs on the training split of ImageNet (Howard, 2019) using a NVIDIA A100 (40GB) GPU. We follow the same cosine annealing learning rate scheduler from Hesse et al. (2021) with an initial learning rate of 0.1 and use a mixup interpolation proposed in Zhang et al. (2017a) with an interpolation strength $\alpha = 0.7$ and a batch size of 256, where each epoch takes approximately 40 minutes to complete. Our model is trained with SGD using a 0.9 momentum, and a 1e-4 weight decay. To remove the bias, we mainly employ the Fixup approach and utilize code from two sources [2][3], which are released under the Apache 2.0 and BSD 3-Clause licenses, respectively. In the CIFAR-100 experiment, we utilize a variant of ResNet18 in which we intentionally omitted the Batch Normalization layers. We train the models with a batch size of 128 for 40 epochs, using the same other hyperparameters.

In accordance with the information provided in Table 1, our FCN and CNN models are trained using the following configurations: The FCN model undergoes 20 epochs of training on the training split of the MNIST dataset, utilizing a learning rate of 0.01 and a batch size of 512. On the other hand, the CNN model is trained for 20 epochs on the training split of the Fashion-MNIST dataset, employing a learning rate of 0.01 and a batch size of 128. Both models are trained with Adadelta optimizer.

2. https://github.com/visinf/fast-axiomatic-attribution
3. https://github.com/hongyi-zhang/Fixup

Appendix D. Interpolation Robustness Examples

In this section, we show some interpolation of images from MNIST and CIFAR10 datasets. Based on our experiments, we observe that sharing the same neural activation pattern is a more stringent condition than having the same prediction label. To enforce this constraint, we opt for smaller networks with fewer neurons. However, this can lead to lower accuracy. To provide a visual illustration of the result, we present some examples of correctly predicted images in Figure 4, and examples of incorrectly predicted images in Figure 5.

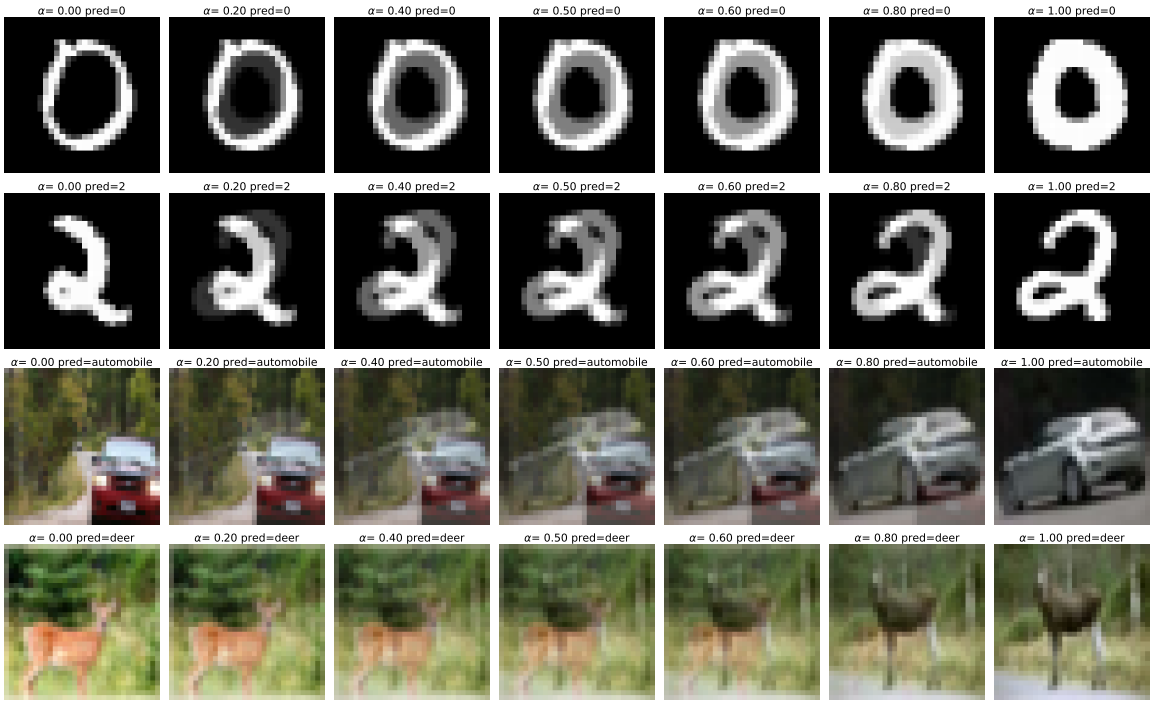


Figure 4: The left-most and right-most images are from the original MNIST (first five rows) and CIFAR10 (the last five rows) dataset, whereas synthesized/interpolated images are in the middle. For instance, the middle image in the first row is generated by adding ($\alpha = 0.5$) times the left image to ($1 - \alpha$) times the right image. The interpolated images have the predictions same as the ground truth.

Figure 5: The left-most and right-most images are from the original MNIST (first five rows) and CIFAR10 (the last five rows) dataset, whereas synthesized/interpolated images are in the middle. For instance, the middle image in the first row is generated by adding ($\alpha = 0.5$) times the left image to ($1 - \alpha$) times the right image. The synthesized/interpolated images have the predictions different from the ground truth.