ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Departamento de Engenharia de Computação e Sistemas Digitais

PCS 5022 - Redes Neurais e Aprendizado Profundo

Prof. Artur Jordão

1 Fundamentos Básicos de Machine Learning

1. O código abaixo apresenta um problema teórico grave. Qual? Como resolver? Crie um modelo preditivo (i.e., OLS, SVM, XGBoost) e analise o comportamento da performance preditiva do modelo antes e após resolver o problema.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

n_samples = 1000
n_features = 2
n_classes = 5
X = np.random.rand(n_samples, n_features)
y = np.random.randint(0, n_classes, size=n_samples)
y = np.eye(n_classes)[y]

mean = np.mean(X, axis=0)
std = np.std(X, axis=0)
X = (X - mean) / std

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33,
random_state=42)

model = ...
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

accuracy_score(y_test, y_pred)
```

Código 1: Exercício 1

2. Gere dados aleatórios com dimensão $n \times m$ para k categorias (classes) e $m = 2$. A partir dos dados gerados realize as seguintes tarefas. (i) Calcule a média das amostras; (ii) Calcule a média das amostras de cada classe; (iii) Para cada classe, identifique qual a amostra mais distante da amostra média da classe. Expresse formalmente a solução dos itens (i)-(iv).

3. Gere dados aleatórios com dimensão $n \times m$, com $m = 2$. A partir desses dados mostre o espaço de características, destacando a amostra média geral e a amostra média por classe.

4. Faça o mesmo que o exercício anterior, porém agora utilize $m > 4$. Como mostrar o espaço de características quando m é arbitrariamente grande?

5. Um problema envolvendo a solução analítica da regressão linear, $(X^T X)^{-1} X^T Y$, é que os dados de treinamento não podem apresentar mais features (m) do que amostras (n) – *the sample size problem*. Como resolver esse problema sem utilizar a otimização de *gradient descent*?

6. Suponha um cenário hipotético em que só podemos aprender um modelo usando um determinado número de amostras k, onde $k \ll n$. Como lidar com essa restrição evitando uma solução trivial de amostrar exemplos aleatoriamente?
   Leitura recomendada para o exercício: [21]

2 FeedForward Networks

1. Assuma dois valores inteiros n e k. A partir desses valores, crie uma rede MLP com n camadas, onde a i-ésima camada ($i \in \{1, 2, 3, \ldots, n\}$) possui k^i neurônios.

2. ($\star$) Uma similaridade entre projeções lineares (ex., predição de uma regressão linear ou transformação via *Principal Components Analysis* – PCA) e uma rede neural com somente uma camada é que podemos formular a predição como $xW^T + b$. Tecnicamente, podemos utilizar essa similaridade para modelar predições de modelos lineares como redes neurais. Como? Tente realizar esse processo para uma projeção linear simples (ex. PCA). Observe como a matriz de projeção é armazenada: matriz coluna – $k \times 1$ – ou matriz linha – $1 \times k$. Note também as possíveis transformações que podem ser realizadas nos dados antes de projetá-los. O código abaixo ilustra com utilizar o PCA disponível no sklearn.

```
from sklearn.decomposition import PCA
from sklearn.datasets import make_classification
import numpy as np

X, y = make_classification(n_samples=10000, n_features=3000...)

pca = PCA(n_components=2)
pca.fit(X)
X_latent = pca.transform(X)
w = pca.components_
...
```

Código 2: Exercício 2

3. Podemos dizer que modelos superparametrizados podem classificar dados com rótulos aleatórios. Desenvolva um setup experimental para demonstrar (mesmo que parcialmente) essa afirmação. Tal característica corresponde a uma propriedade positiva ou negativa desses modelos?
   Leitura recomendada para o exercício: [32, 22]

3 Basics Hyperparameters

1. Formule o grafo computacional das seguintes expressões: $y = 2 * (a - b) + c$; $y = x^T w + b$ e $y = (\frac{x-\mu}{\sigma}) * \gamma + \delta$.

2. A partir do grafo computacional abaixo, indique qual a derivada parcial de $\frac{\partial g}{\partial b}$ (isto é, como alterações em b afetam g).

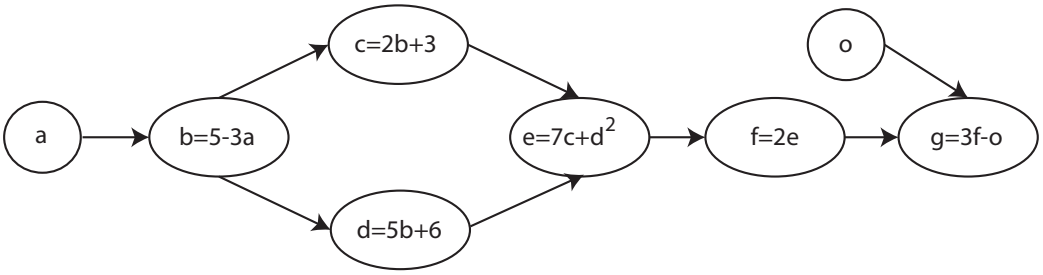

Figure 1: Grafo computacional para o Exercício 2

3. (⋆) Proponha uma técnica *learning rate scheduler*.

4. (⋆) Proponha uma função de ativação.
Leitura recomendada para o exercício: [27].

5. (⋆) Proponha uma função de erro customizada. A função de erro pode ser uma variação de alguma função existente (ex. adicionando um termo de penalização) ou combinação de operadores matemáticos primitivos.
Leitura recomendada para o exercício: [17, 18].

6. Elabore uma técnica para explorar um espaço n dimensional de hiperparâmetros (ex. *grid search*). Seu espaço deve conter, pelo menos, os seguintes hiperparâmetros: (i) otimizador, (ii) inicialização e (iii) *learning rate* inicial. Em cada eixo do espaço (isto é, cada hiperparâmetro) deve existir pelo menos duas opções. Visualizando essa questão da perspectiva de *feature space*, seria possível predizer a performance preditiva do modelo para valores de hiperparâmetros não observáveis (dica: pense como seria a composição das matrizes X e Y)?

4 Deep Learning

1. Frequentemente em *Deep Learning*, a quantidade de parâmetros é um indicativo da capacidade de expressividade do modelo. A partir dessa observação, uma forma de controlar a capacidade de um modelo é aumentar o número de neurônios ou a quantidade de camadas (paradigma *we need to go deeper*). Assuma um valor inteiro n indicando a quantidade de

neurônios de duas camadas ocultas compondo uma rede neural. Construa outra arquitetura distribuindo esses n neurônios em k camadas. Por exemplo, a partir de uma rede com $n = 2048$ (largura $\{2048, 2048\}$) podemos criar outra com o mesmo número de neurônios usando $k = 4$ ($\{1024, 1024, 1024, 1024\}$) ou $k = 8$ ($\{512, 512, 512, 512, 512, 512, 512, 512\}$). Qual dos dois modelos é mais eficiente em termos de latência[1], armazenamento e número total de parâmetros?
Leitura recomendada para o exercício: [28, 8, 2]

2. ($\star$) Podemos entender a transformação aplicada por uma camada i como a representação interna dessa camada para os dados. A partir dessa descrição, como podemos mensurar a similaridade entre representações internas de duas camadas distintas da mesma rede? E de redes neurais diferentes? O código abaixo sumariza como obter a representação interna de uma camada i usando um modelo pré-definido.
Leitura recomendada para o exercício: [15, 24, 25, 4]

```
model_ = keras.models.Model(model.input,
model.get_layer(index=i).output)
internal_representation = model_.predict(X)
```

3. Combine a representação interna de diferentes camadas e aplique essa combinação ($X = \bigcup_{i=1}^{L} X_i$) a um classificador simples.

4. Considerando uma rede neural qualquer, elabore um experimento para ilustrar a separabilidade fornecida pela representação interna das diferentes camadas compondo o modelo.

5. De acordo com o trabalho de Zhang et al. [33], as camadas em modelos profundos podem ser categorizadas em robustas e críticas. Camadas robustas são aquelas que, após o treinamento, podem ser reinicializadas a sua inicialização original sem degradar (ou degradando marginalmente) a habilidade preditiva do modelo. Camadas críticas, por outro lado, são sensível a reinicialização. Elabore um setup experimental simples para demonstrar essa observação. Ao final, mostre quais foram as camadas críticas e as robustas. Observação importante: as conclusões de Zhang et al. [33] são aplicáveis às arquiteturas residuais que serão abordadas posteriormente no curso.
Leitura recomendada para o exercício: [33, 23]

5 Regularization

1. Construa uma arquitetura de rede neural simples e a regularize-a utilizando o mecanismo de ensemble. Defina formalmente a estratégia de ensemble adotada.

[1] Latência refere-se ao tempo que um modelo leva para predizer a resposta a partir de uma amostra ou conjunto de amostras (tempo de *foward*).

2. Períodos críticos correspondem a um fenômeno relacionado à efetividade da regularização na dinâmica do treinamento. De acordo com trabalhos anteriores, tal fenômeno ocorre nas épocas iniciais de treinamento. A partir de uma rede neural qualquer, elabore um setup experimental para inspecionar se períodos críticos emergem durante o treinamento dessa arquitetura.
   Leitura recomendada para o exercício: [6, 13, 14]

3. Conforme visto em aula, estudos mostraram a possibilidade de aprender somente os parâmetros das camadas de Batch Normalization. Elabore um treinamento para demonstrar a eficácia dessa estratégia (isto é, se um modelo treinado seguindo essa ideia obtém habilidade preditiva não trivial).
   Leitura recomendada para o exercício: [5, 1]

6 Convolutional Networks

1. Selecione uma rede neural convolucional (por exemplo, a partir do keras applications) e, a partir desta rede, escolha n camadas para realizar extração de *features*. Para cada camada $i \in n$, projete suas respectivas *features* em um espaço de duas dimensões (ex., usando PCA ou outro método de redução de dimensionalidade). Em seguida, analise se alguma camada proporciona uma melhor separação dos dados.

2. Um dos maiores gargalos computacionais envolvendo redes convolucionais está relacionado à resolução espacial (largura $\times$ altura – $W \times H$) da imagem de entrada. Interessantemente, estudos confirmaram a possibilidade de adaptar uma rede pré-treinada para realizar a inferência (*forward*) em uma resolução diferente da utilizada durante o treinamento. Por exemplo, Touvron et al. [29] observou que é possível treinar uma rede convolucional em determinada escala e aplicar uma escala maior somente nas imagens de teste; desta forma, reduzindo notavelmente o tempo de treinamento/*fine-tuning*.

   Formalmente, seja $\mathcal{F}(\cdot^{224\times224}, \theta)$ uma rede convolucional treinada a partir de amostras com resolução 224×224. Proponha uma solução que permita utilizar θ sem qualquer alteração para predizer amostras de resoluções arbitrárias ($x \in \mathbb{R}^{W\times H}$ – $\mathcal{F}(\cdot^{W\times H}, \theta)$). Qual componente da arquitetura possibilita a solução? Observe que o problema não corresponde a alterar a resolução da entrada. Para facilitar a implementação utilize a arquitetura ResNet50 disponível no github da disciplina e os pesos treinados no ImageNet disponíveis neste link.
   Leitura recomendada para o exercício: [29]

3. ($\star$) Seguindo o exercício acima, proponha um modelo para predizer qual a melhor resolução para realizar o *forward* de uma imagem x em uma rede convolucional. Por exemplo, considere um modelo $\mathcal{P}(x, \cdot)$ que estima qual a melhor resolução para predizer x a partir de um conjunto de resoluções pré-definidas (y). O que poderia ser considerado como *melhor resolução?*
   Leitura recomendada para o exercício: [31]

4. Crie uma rede convolucional que realize a classificação de imagens. Sua arquitetura pode conter **apenas** camadas convolucionais, *padding* e (opcionalmente) operações de

reshape/flatten. A arquitetura **não** deve conter camadas *fully-connected*.
Leitura recomendada para o exercício: [11]

5. Utilize a técnica de *loss landscape* para visualizar o espaço de parâmetros de uma rede neural (convolucional ou não) treinada. O código para gerar o *loss landscape* está disponível no github da disciplina.
Leitura recomendada para o exercício: [19]

7 Residual Networks

1. Conforme visto em aula, para facilitar a construção de redes complexas e profundas uma prática comum é desenvolver *building blocks* e, então, replicá-los para obter a arquitetura final. Proponha um *building block* com no mínimo 6 componentes. Feito isso, construa uma arquitetura de rede neural repetindo o *building block* proposto n vezes.

2. Faça o mesmo que o exercício anterior, porém, agora insira *skip connections* **entre** *building block*.
Leitura recomendada para o exercício: [9]

3. A arquitetura abaixo apresenta o problema de *miss matching* que ocorre quando a operação de soma *point-wise* entre a entrada (*shortcut*) e saída da camada residual não possuem correspondência. O que deveria ser alterado para que a arquitetura funcionasse corretamente?

```
...
d = 2
inputs = keras.layers.Input(shape=X.shape[-1])
x = inputs
x = layers.Dense(d, activation='relu')(x)

for _ in range(n_blocks):
 shortcut = x
 x = layers.Dense(d, activation='relu')(x)
 x = layers.Dense(d * 2, activation='relu')(x)
 x = layers.Dense(d * 4, activation='relu')(x)
 x = layers.Dense(d * 2, activation='relu')(x)
 x = x + shortcut
 d = d * 2
 x = layers.Dense(d, activation='relu')(x)

outputs = layers.Dense(n_classes, activation='softmax')(x)
model = tf.keras.Model(inputs, outputs)
```

Código 3: Exercício 3

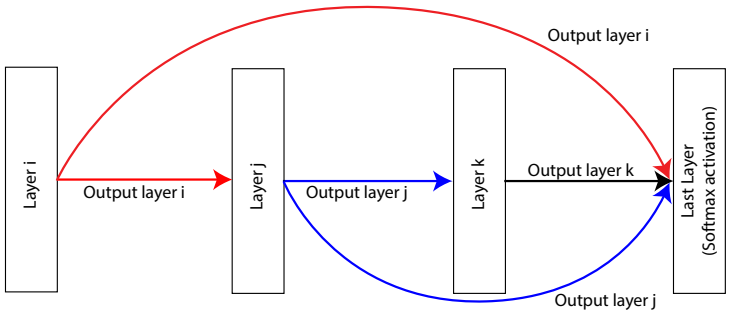

Figure 2: Exemplo de arquitetura para o Exercício 4. Formalmente, a entrada para a última camada L é $\bigcup_{i=1}^{L-1} y_i$, onde y_i corresponde à saída da ith camada.

4. Uma característica inerente de redes residuais é que elas possibilitam lidar com o *vanishing gradient* uma vez que o gradiente chega tanto em $f(x)$ quanto em x (lembre-se que em redes residuais $y_i = f_i(x) + x$). Outra forma de lidar com o *vanishing gradient* é conectar diretamente todas as camadas ocultas à camada de saída (ex., camada *fully connected* com a ativação *Softmax*). Proponha uma arquitetura para lidar com o *vanishing gradient* agregando a representação das camadas por meio da operação de concatenação (ex.:). A Figura 2 ilustra como seria tal arquitetura. Por simplicidade, você pode considerar uma rede MLP simples. Após elaborar sua arquitetura, pense em quais limitações a operação de concatenação mitiga em relação à operação de soma (*point-wise*) proposta inicialmente para as redes residuais.
   Leitura recomendada para o exercício: [12]

5. ($\star$) Uma característica interessante das redes residuais é que elas permitem que o fluxo de informações (representação) siga caminhos diferentes através da rede. Desta forma, as camadas podem depender fracamente umas das outras. De acordo com trabalhos da literatura, essa característica permite *apagar* algumas camadas durante o processo de *forward* sem comprometer a habilidade de predição da rede. Formalmente, considere y_i a saída da ith camada de uma rede residual, $y_i = f_i(y_{i-1}, \theta_i) + y_{i-1}$. A partir desta descrição, durante o *forward* podemos zerar os pesos de θ_i levando à seguinte transformação $y_i = 0 + y_{i-1}$, que é equivalente a propagar somente a entrada y_{i-1}. Proponha um setup experimental para encontrar quais camadas residuais não afetam a habilidade preditiva da rede. (Observação: lembre-se de retornar os valores de θ_i ao apagar uma nova camada j)
   Leitura recomendada para o exercício: [30, 3]

8 Data Augmentation

1. Elabore um experimento para verificar se é mais importante introduzir quantidade ou diversidade de amostras durante o treinamento de redes neurais.

2. Proponha uma técnica de aumento de dados fora do domínio de visão computacional.

Verifique a efetividade da técnica e discuta os resultados.

3. (⋆) Uma maneira de lidar com amostras adversariais é gerar e introduzir (*adversarial training*) tais amostras como parte dos exemplos de treinamento. Estudos relativamentre recentes, entretanto, vêm demonstrando resultados positivos ao lidar com cenários adversariais aplicando *data augmentation* nas amostras de teste (processo tipicamente denominado *testing time adaptation*). Por exemplo, Guo et al. [7] evidenciaram um fenômeno denominado *scaled prediction consistency*. A partir desse fenômeno, os autores sugerem aumentar o tamanho das imagens de entrada (x) e verificar uma pontuação de confiança (veja Equação 2 do artigo): se a confiança diminuir, a imagem corresponde a uma amostra benigna; caso contrário, a amostra é adversarial. Liu et al. [20] observaram que modelos infectados por ataques *backdoor* exibem uma robustez à corrupção semântica diferente para amostras com *trigger*. Os autores exploram essa observação para detectar amostras com *triggers*. A partir desses *insights*, proponha uma forma de *data augmentation* no teste para aumentar a robustez adversarial.
Leitura recomendada para o exercício: [7, 10, 20]

4. (⋆) Uma questão prática envolvendo o ataque *Fast Gradient Sign Method* (FGSM) é a sua dependência do cálculo do gradiente em relação à amostra x. Devido à essência de amostras destiladas – possuem toda a informação discriminativa das amostras originais –, uma ideia (acredito que interessante) seria utilizar amostras destiladas ($IPC = 1$) como a imagem base para o cálculo do gradiente. Formalmente, o ataque FGSM seria $x' = x + \epsilon \cdot sign(\nabla_\theta \mathcal{L}(\theta, x_c, \cdot))$, onde x_c corresponde a uma amostra destilada da classe c. Elabore essa versão do ataque FGSM. Note a necessidade de um rótulo no equacionamento e sua solução deve considerar isso. A partir do ataque implementado, as seguintes questões de pesquisa surgem naturalmente. (i) É possível produzir um ataque FGSM agnóstico ao gradiente da amostra x? Isto é, o ataque proposto é efetivo? (ii) Amostras destiladas geradas por um modelo tornam-se transferíveis para outras arquiteturas? Para este exercício, recomenda-se utilizar amostras destiladas pré-calculadas. Para este propósito, o seguinte repositório pode ser útil: link para o github
Leitura recomendada para o exercício: [26, 16]

References

[1] Rebekka Burkholz. Batch normalization is sufficient for universal function approximation in CNNs. In *International Conference on Learning Representations (ICLR)*, 2024.

[2] Mostafa Dehghani, Yi Tay, Anurag Arnab, Lucas Beyer, and Ashish Vaswani. The efficiency misnomer. In *International Conference on Learning Representations (ICLR)*, 2022.

[3] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning (ICML)*, 2021.

[4] Lyndon R. Duong, Jingyang Zhou, Josue Nassar, Jules Berman, Jeroen Olieslagers, and Alex H. Williams. Representational dissimilarity metric spaces for stochastic neural networks. In *International Conference on Learning Representations (ICLR)*, 2023.

[5] Jonathan Frankle, David J. Schwab, and Ari S. Morcos. Training batchnorm and only batchnorm: On the expressive power of random features in cnns. In *International Conference on Learning Representations (ICLR)*, 2021.

[6] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[7] Junfeng Guo, Yiming Li, Xun Chen, Hanqing Guo, Lichao Sun, and Cong Liu. SCALE-UP: An efficient black-box input-level backdoor detection via analyzing scaled prediction consistency. In *International Conference on Learning Representations (ICLR)*, 2023.

[8] Kai Han, Yunhe Wang, Qiulin Zhang, Wei Zhang, Chunjing Xu, and Tong Zhang. Model rubik's cube: Twisting resolution, depth and width for tinynets. In *Neural Information Processing Systems (NeurIPS)*, 2020.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] Zhiyuan He, Yijun Yang, Pin-Yu Chen, Qiang Xu, and Tsung-Yi Ho. Be your own neighborhood: Detecting adversarial examples by the neighborhood relations built on self-supervised learning. In *International Conference on Machine Learning (ICML)*, 2024.

[11] Andrew Howard, Ruoming Pang, Hartwig Adam, Quoc V. Le, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, and Yukun Zhu. Searching for mobilenetv3. In *International Conference on Computer Vision (ICCV)*, 2019.

[12] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] Michael Kleinman, Alessandro Achille, and Stefano Soatto. Critical learning periods for multisensory integration in deep networks. In *Conference on Computer Vision and Pattern Recognition (CVPR*, 2023.

[14] Michael Kleinman, Alessandro Achille, and Stefano Soatto. Critical learning periods emerge even in deep linear networks. In *International Conference on Learning Representations (ICLR)*, 2024.

[15] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning (ICML)*, 2019.

[16] Shiye Lei and Dacheng Tao. A comprehensive survey of dataset distillation. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2024.

[17] Chuming Li, Xin Yuan, Chen Lin, Minghao Guo, Wei Wu, Junjie Yan, and Wanli Ouyang. AM-LFS: automl for loss function search. In *International Conference on Computer Vision (ICCV)*, 2019.

[18] Hao Li, Tianwen Fu, Jifeng Dai, Hongsheng Li, Gao Huang, and Xizhou Zhu. Autoloss-zero: Searching loss functions from scratch for generic tasks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[19] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Neural Information Processing Systems (NeurIPS)*, 2018.

[20] Xiaogeng Liu, Minghui Li, Haoyu Wang, Shengshan Hu, Dengpan Ye, Hai Jin, Libing Wu, and Chaowei Xiao. Detecting backdoors during the inference stage based on corruption robustness consistency. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[21] Sepideh Mahabadi and Stojan Trajanovski. Core-sets for fair and diverse data summarization. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[22] Pratyush Maini, Michael Curtis Mozer, Hanie Sedghi, Zachary Chase Lipton, J. Zico Kolter, and Chiyuan Zhang. In *International Conference on Machine Learning (ICML)*, 2023.

[23] Wojciech Masarczyk, Mateusz Ostaszewski, Ehsan Imani, Razvan Pascanu, Piotr Miłoś, and Tomasz Trzcinski. The tunnel effect: Building data representations in deep neural networks. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[24] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations (ICLR)*, 2021.

[25] Thao Nguyen, Maithra Raghu, and Simon Kornblith. On the origins of the block structure phenomenon in neural network representations. *Transactions on Machine Learning Research*, 2022.

[26] Jonathan Peck, Bart Goossens, and Yvan Saeys. An introduction to adversarially robust deep learning. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2024.

[27] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. In *International Conference on Learning Representations (ICLR)*, 2018.

[28] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2019.

[29] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[30] Andreas Veit, Michael J. Wilber, and Serge J. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Neural Information Processing Systems (NeurIPS)*, 2016.

[31] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[32] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

[33] Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *Journal of Machine Learning Research*, 2022.