Rethinking Data Ordering:
Investigating Alternative Strategies for Enhanced Performance in Computer Vision Tasks

Julia Lange, Joshua Kovac, Rory Little, Ryan Hollobon, and Scott Wehrwein
{langed, kovacj, littler4, hollobr, wehrwes}@wwu.edu
Computer Science Department
Western Washington University

***Abstract*—Machine learning models learn through iterative exposure to numerous examples. Data ordering, the sequence of presenting these examples, is crucial for training, particularly in computer vision tasks. Despite Random ordering being the standard, we investigate alternatives for more efficient implementations, such as in video data. We investigate various data ordering techniques' impacts on different computer vision tasks and architectures, aiming to identify more effective methods.**

We conduct a series of experiments evaluating AutoEncoders and Residual Networks on image reconstruction and classification using datasets with varying levels of diversity. We assess performance using common Data Ordering strategies including Sequential ordering and Random ordering, as well as several novel ordering strategies including Varying Speed Sequential ordering, and a Randomized Subbatch ordering.

Our results reveal that alternative Data Ordering strategies are competitive with, and in certain cases outperform, random access. These findings indicate potential for improvement over the conventional random access method and emphasize the importance of considering data ordering when designing and training computer vision models.

I. Introduction

Modern machine learning research typically centers on three fundamental aspects: the dataset, the model, and the loss function. While these are critically important, one vital element often gets overlooked – the methodology used to sample from the dataset. Researchers tend to rely on Random ordering, a prevalent practice supported by substantial empirical evidence. However, this assumption might not hold in all cases. This is compounded by the fact that Random ordering can be challenging to implement efficiently.

This issue often emerges in certain computer vision tasks, such as when sampling frames from a video. In this context, the implementation of Random ordering can become computationally expensive due to the encoding of video data. As a result, Sequential ordering – the direct opposite of Random ordering – might emerge as a more appealing strategy, due to its relative ease of implementation.

Simultaneously, there's another scenario to consider. Imagine a dataset where a large majority of the samples are strikingly similar, but are evenly distributed throughout the dataset. In this case, using Random ordering could lead to suboptimal results. The rationale behind this is that Random ordering could end up feeding a substantial number of these similar samples sequentially into the model. The model's performance would then degrade due to being supplied repetitive data.

These situations prompt us to reconsider the dominance of Random ordering, and investigate if other strategies can come close to or beat Random ordering. Exploring other Data Ordering strategies could lead to more diverse and informative samples, improving the efficiency and efficacy of our models.

Our research aims to investigate alternative data ordering strategies by considering various datasets, models, tasks, and new data ordering strategies. We aim to compare the performance of these alternatives using both quantitative and qualitative metrics.

We assess permutations of AutoEncoders for reconstruction, and Residual Networks for classification on ImageNet [5], ADE20K [7], and a custom dataset, with five different data ordering strategies. Our investigation covers widely known data ordering strategies such as random and sequential, as well as underexplored or novel techniques such as Randomized Sequential Chunks, Tortoise-Hare Read Heads, and Subbatched Random Access.

Our experiments reveal that a well-chosen Data Ordering strategy can more effectively exploit a dataset's diversity. We demonstrate scenarios where random ordering is comparable to alternative methods, challenging the conventional wisdom. We introduce two novel data ordering strategies – the Tortoise-Hare Sampler, and the Subbatched Multiframed Random Access Sampler. Interestingly, we uncover dataset, task combinations where the choice of Data Ordering strategy seems to make little difference, yielding equivalent results for all our Data Ordering strategies random ordering. This unexpected result further underscores the importance of reconsidering and examining our choice of Data Ordering strategy when building machine learning models.

II. Background and Related Work

In the field of machine learning, the ordering of data for training models holds substantial significance. A notable component of this process is Stochastic Gradient Descent (SGD). SGD plays a vital role in traversing the parameter space of the model and converging on an optimal solution. Due to how SGD weights samples as they are input, early data has a huge impact on the resulting approximation. If we show a model first pictures of dogs, then of cats, the model could get exceptionally good at performing its task on dogs, overfitting to the specific features of dogs. When the model then sees cats it has to work extra hard to forget many of the

less generalizable aspects it learned from dogs. This effect has been measured empirically by Meng et al [4].

Because of this behavior we often think of Sequential ordering as the worst method. Here, when we talk about Sequential, we assume some order for the dataset. Often a video dataset will be ordered by time, or a classification dataset ordered by classes. If our dataset is a randomized set of images, then Sequential ordering is meaningless.

Other research has gone into developing data ordering strategies based on the way humans learn. Bengio et al [1] consider Data Ordering strategies they call "Curriculums" which attempt to go from easier to learn examples to harder more complicated ones. These concepts are successful and perform better than Random ordering in certain cases. Weinshall et al [6] expand on this research by using pre-trained networks to rank training examples by difficulty before feeding them to a new model.

A paper by Chang et al [2] further explored the concept of using the loss, and other aspects of training to improve the model, specifically with a focus on Stochastic Gradient Descent. All of these papers focus on broader machine learning impacts, and less on computer vision. Additionally there is a lack of focus on the practical considerations of the underlying data structures, such as when dealing with Video.

III. Experimental Setup

A. Model Architectures

The AutoEncoder was chosen as a versatile, and basic structure to experiment on. Hopefully providing a baseline for convolutional models. This model was used for image reconstruction, due to the simplicity of the model and task.The encoder comprises three layers, each made up of a 3x3 convolution with padding of 1, into ReLU into a 2x2 MaxPool with stride 2. The convolutions change the number of channels first from 3 to 8, then 8 to 12, and finally 12 to 16. The decoder similarly comprises three layers which calculate a 2x2 2D transposed convolution, then ReLU. Except the final layer, where the ReLU is swapped out for a Sigmoid. Each layer changes the number of channels in the reverse order from the Encoder (16 to 12, 12 to 8, 8 to 3).

The Residual Network implemented is a ResNet V1.5 [3]. Both ResNet18 and ResNet50 were used, though both were constrained to 10 epochs. The relatively low number of layers, and epochs were chosen due to the large number of experiments to run relative to the amount of compute power available. This choice was made with the assumption that any differences observed between 18 and 50 layers would be seen in larger layers, such as 100 layers. ResNets were only used for classification tasks, as any reconstruction trends will be captured by the faster to train AutoEncoder.

B. Datasets

ImageNet is filled with a million images of striking diversity, which has caused ImageNet to be used for a large diversity of models. By including ImageNet we hope to provide further insight into the countless models built off of this Dataset. We defined the Sequential ordering of ImageNet as the classes ordered by their folder name, and then every image belonging to that class ordered by its file name. We have no reason to believe the folder or image names were assigned in any order besides randomly.

A custom dataset composed of long-duration webcam streams of diverse outdoor scenes we will refer to as WWU Webcam was chosen due to the atypical nature of the dataset, as well as the massive amount of repetition. The dataset contains still camera footage from various scenes, such as a construction site, a norwegian dock and an ice skating rink.

The dataset is massive, with a single scene containing over a hundred days worth of footage, or around 2.7 billion images. While this scene is 200 times larger than ImageNet it is worth highlighting that most of the frames are the same as each other, with nothing changing. Use of this dataset was limited to two scenes, Rane (the construction site) and Geiranger (the norwegian dock), with 48 hours of footage, only considering every 30th frame, which cuts the size of the data to a manageable 200 thousand images per scene. The Sequential Ordering of a WWU Webcam scene is considered the chronological order the video plays in.

ADE20K was chosen as a middle ground between these two datasets. This means we do not consider the incredibly useful semantic masks of ADE20K. Instead the 10 general classifications, and the similar setting of rooms, landscapes, and buildings is used as an inbetween in variance between the incredibly varied ImageNet, and the incredibly repetitive WWU Webcam. The structure of the ADE20K dataset is a list of folders for broad categories. Within each of these broad categories are a list of subcategories, which contain a list of images where each image has a unique number. Motivated by this the Sequential ordering for ADE20K is the alphabetically sorted categories, each containing the alphabetically sorted subcategories, and then the images ordered by their unique number.

C. Data Ordering Strategies

Five Data Ordering Strategies are considered. Those are Sequential Ordering, Random Ordering, Random N-Chunk Ordering, Tortoise-Hare Ordering, and SBMFRAW Ordering. Examples of the outputs of these orderings are shown in Figure 1.

1) Random and Sequential: Random ordering is considered as an upper bound based on many justified assumptions about Data Ordering. Random orderings are achieved by randomly sampling from the input without replacement. By contrast Sequential orderings are our lower bound. The Sequential ordering varies depending on the context, and so is defined per Dataset.

2) Random N-Chunk: Motivated by hardware limitations, such as how much data you could fit in VRAM, and as a consistent middle ground between Sequential and Random Ordering is Random N-Chunk ordering. Random N-Chunk ordering is the strategy of choosing some chunk size N, and then defining the first N elements of the Sequential Ordering

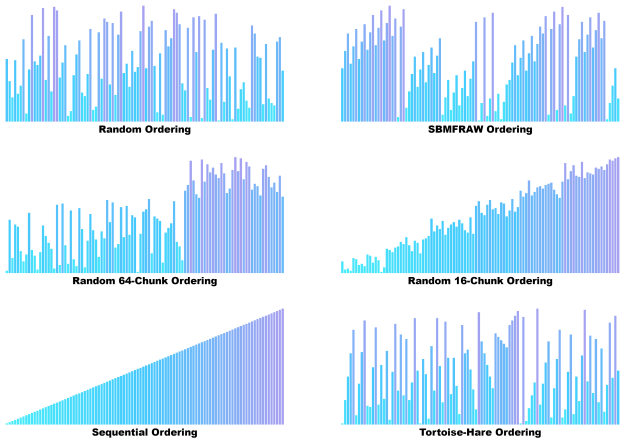

Fig. 1: Line charts of a 100 entry dataset sampled with different Data Orderings. From left to right top to bottom; Random Ordering, SBMFRAW Ordering using subbatch list of $[0, 2, 4, 8]$ and multiframe list of $[-15, 0, 15]$, Random 64-Chunk Ordering, Random 16-Chunk Ordering, Sequential Ordering, Tortois-Hare Ordering using step size 7 with 5 read heads.

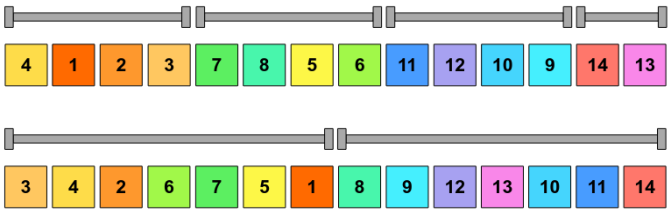

Fig. 2: A list of 14 numbers sampled with Random N-Chunk, with 4-Chunk above and 7-Chunk below.

as a chunk. We then randomly sample from that chunk. Once the chunk is exhausted we grab the next N elements of the Sequential Ordering and repeat. If we get to the last M elements of the Sequential Ordering and $M < N$, then we take a chunk of size M and randomly sample from it instead. This behavior can be seen in Figure 2.

3) Tortoise-Hare: When reading from a video sequential reads are much faster than random or backwards reads. Tortoise-Hare ordering seeks to take advantage of that fact, by

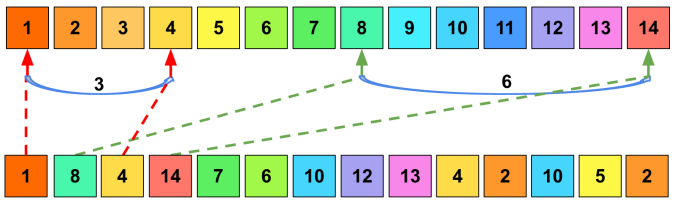

Fig. 3: A list of 14 numbers sampled from Tortoise-Hare, with two read heads and a step size of 3.

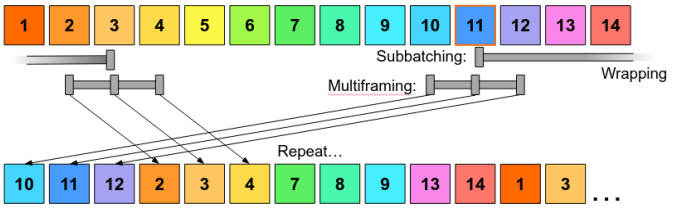

Fig. 4: Sampling from a list of 14 numbers with SBMFRAW. With a subbatch list of [0,6] and multiframe list of [-1,0,1]. The number 11 is randomly chosen as the first random sample.

creating a pseudo-random output using sequential reads. We achieve this by having multiple read heads moving at different speeds, wrapping at the edges of our data.

We choose R read heads, and a step size S. We give each read head a unique id i from 0 to R. Assume D is the size of the dataset. We then evenly distribute the read heads across the dataset, by setting their position to $i * \frac{D}{R}$. Each read head is then assigned a "speed" which is equal to their $(i+1) \cdot S$ This means that the 1st read head is half as fast as the 2nd read head, and a third as fast as the 3rd read head. We then iterate through each read head, getting the value at its position in the dataset, before moving it forward based on its speed. We repeat this iteration over all read heads $\frac{D}{R}$ times. The procedure can be seen visually in Figure 3.

The Tortoise-Hare ordering is deterministic, and does not employ randomness. Additionally, it is possible for it to repeat elements, and skip elements. When using relatively prime values for R and S the strategy provides a varied and pseudo random sample, which requires only forward reads, and with wrapping.

4) SBMFRAW: In the WWU Webcam dataset our data is semi-structured. Most frames will be the same objects and background but during a different time of day. This means that Random Ordering tends to still sample very similar images when used. Motivated by this we created Subbatched Multi-framed Random Access Wrapped ordering (SBMFRAW). The Goal of SBMFRAW is to get a variety of intervals between two frames. When randomly selecting two frames, the distance between them in a sequential ordering is usually very high. SBMFRAW attempts to lower the average interval between frames by adding in smaller sequential intervals alongside the large random intervals.

SBMFRAW accomplishes this by first selecting a random frame, and then get nearby frames sequentially, we call each of these nearby frames, and the randomly selected one subbatches. After that we get frames around each one of these subbatches, hence the multiframe part. Originally these multiframes would all be fed to a model at once. Despite none of the models we're testing utilize multiframed input, we've kept the ordering the same.

In order to accomplish the above, SBMFRAW is provided a subbatch list S and a multiframe list M. The subbatch list tells us what frames around our randomly sampled frame to

	Sequential	Random	Random 64-Chunk	SBMFRAW	Tortoise-Hare
ImageNet	$\mathbf{3.975 \cdot 10^{-3}}$	$4.040 \cdot 10^{-3}$	$4.020 \cdot 10^{-3}$	$4.213 \cdot 10^{-3}$	$4.088 \cdot 10^{-3}$
ADE20K	$4.577 \cdot 10^{-3}$	$\mathbf{3.996 \cdot 10^{-3}}$	$4.075 \cdot 10^{-3}$	$4.286 \cdot 10^{-3}$	$4.703 \cdot 10^{-3}$
WWU Webcam	$1.587 \cdot 10^{-1}$	$1.653 \cdot 10^{-1}$	$1.512 \cdot 10^{-1}$	$\mathbf{1.484 \cdot 10^{-1}}$	$1.561 \cdot 10^{-1}$

TABLE I: AutoEncoder end of training mean squared error on validation set

	Sequential	Random	Random 16-Chunk	Random 64-Chunk	Random 128-Chunk	SBMFRAW
ImageNet, 18 Layers	40.727	**15.229**	46.721	56.161	45.929	24.243
ImageNet, 50 Layers	61.805	73.153	170.995	81.776	539.274	**41.459**

TABLE II: ResNet18, and 15 end of training cross entropy loss on validation set

	Sequential	Random	Random 16-Chunk	Random 64-Chunk	Random 128-Chunk	SBMFRAW
ImageNet, 18 Layers	$1 \cdot 10^{-3}$	$\mathbf{1.02 \cdot 10^{-3}}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$0.74 \cdot 10^{-3}$
ImageNet, 50 Layers	$1 \cdot 10^{-3}$	$1.1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$\mathbf{1.12 \cdot 10^{-3}}$

TABLE III: ResNet18, and 15 end of training accuracy on validation set

use, which creates a list of subbatches. The multiframe list then tells us what frames around each subbatch to use. Specifically, we select some random frame index c. We then generate the cores of our subbatches $S' = \{(s + c)|s \in S\}$. If any $s + c$ is larger than the length of our dataset, or smaller than 0 we wrap it around to the start or end of the dataset respectively.

Now we have the cores of our subbatches. We then generate our subbatches by generating $B = \{(b + m)|b \in S', \text{and} m \in M\}$. This gives us a list of subbatches, which we sample in order. Once we finish sampling from these subbatches we choose a new random data point and repeat the process. One step of the process can be seen in Figure 4.

It is worth noting the SBMFRAW can repeat entries, and almost definitely will, since SBMFRAW provides the model with "the length of the dataset times the length of the subbatch list times the length of the multiframe list" samples. For our implementation we limit the number of samples by only sampling once for every item in the dataset.

D. Evaluation Metrics

We compare the different models trained using their loss functions throughout training, as well as in a validation step. The loss function used for image reconstruction is mean squared error. We also inspect sample outputs from the model for qualitative anaylsis. For classification a cross entropy loss is used to compare classes. Additionally, an accuracy is computed on the validation step, which is the number of correctly predicted examples divided by the total number of examples.

IV. Results

A. Image Reconstruction

When training our AutoEncoder on ImageNet we found that the Data Ordering strategy didn't matter. Both training loss and validation loss moved down at similar rates for all Data Ordering Strategies, though Sequential and Random 64-Chunk moved down more consistently and slower than other methods. The final results in Table I show Sequential as performing the best, though the difference is small minuscule between all of them except SBMFRAW which seems to perform worse.

The task does not seem so easy that all of the Data Ordering strategies trivially learned it, leading them to all learn at the same rate. We can tell this if we look at Figure 5. It is clear that all the models are performing well, though the fine details on all of them are not there. We conclude that this means that the Data Ordering strategy did not matter in this case.

When training our AutoEncoder on ADE20K, Data Ordering seemed to make an impact. The Validation Loss had a clear hierarchy, with Random ordering performing the best, while Sequential and Tortoise-Hare performed the worse, again reference Table I.

When training our AutoEncoder on WWU Webcam, Data Ordering again appears to make an impact, though it is worth noting that the Validation Loss values are quite high. We can see Validation Loss for the Norwegian port scene in labeled "WWU Webcam" in Table I. Here we can see that SBMFRAW performed the best while Random performed the worst. Subsequent runs of SBMFRAW and Random were performed and the trend stayed, though it was often times less pronounced. The trend was also less pronounced in the construction scene, we hypothesize this is due to the construction scene being more varied, and less repetitive than the Norwegian port scene.

B. classification

The ResNet18 training on ImageNet showed a dependence on Data Ordering. Due to limited compute resources this experiment was not run as long as we would have liked, but the results still appear quite clear. Table II shows the validation loss. The loss is quite high since it is early on in training, but only the Random and SBMFRAW are managing to make progress in training, while other strategies are getting worse with time.

This is further exposed in the accuracy in Table III. Where Random and SBMFRAW are progressing and changing accuracy, while every other strategy is getting only one classification correct out of one thousand. When looking at what the models are actually guessing, all of the models

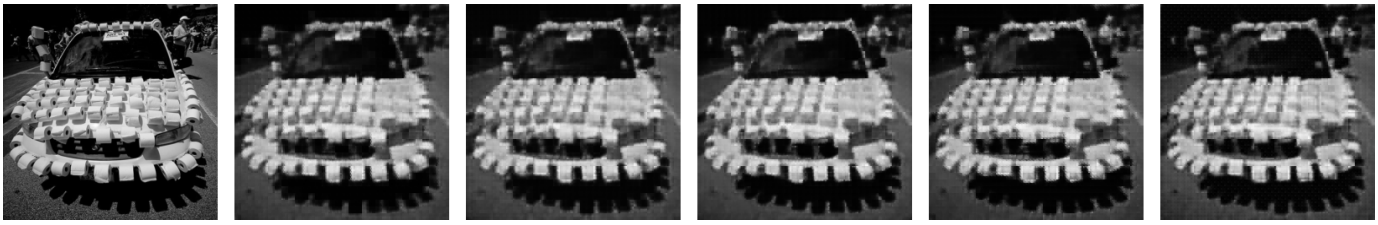

Fig. 5: images produced by the AutoEncoder trained on ImageNet with different Data Ordering strategies. From left to right; Original Image, Sequential, Random, Random 64-Chunk, SBMFRAW, Tortoise-Hare.

except SBMFRAW and Random are guessing the same thing consistently, while Random and SBMFRAW are attempting to guess the actual answer.

This trend continues in the ResNet50 training on ImageNet. Though here the validation loss looks much different. In Table II it looks like not enough time has elapsed for the validation loss to go down by a meaningful amount. We still see Random and SBMFRAW lower than most of the other Data Ordering strategies, but Sequential being with them seems to indicate it isn't significant.

We see the same pattern showing up in accuracy for ResNet50 as we did ResNet18, with SBMFRAW and Random learning while other Data Ordering Strategies repeat the same guess. We can see this behavior in Table III.

V. Discussion, Limitations, and Future Work

A. Discussion

Based on these results we conclude that the most important factor for Data Ordering is the variety between examples shown. Consider ImageNet with a Sequential ordering. The first class we see are Sharks, but within these photos of sharks are a variety of different positions, orientations, and angles of sharks. Because of this when training image reconstruction with our AutoEncoder, the model still has a lot of variety to work with and learn.

We can compare this to our ResNet classification results. For the Sequential ordering, the first chunk of images are sharks, meaning that all the models that only see sharks at the beginning get stuck in a rut they are unable to get out of, and consistently guess the same class.

This trend holds up when moving on to our image reconstruction on ADE20K. The variety in the images is now much less, so the models using more Sequential orderings struggle in comparison to the more Random ones.

If this were the whole story we would expect to see the trend continue in WWU Webcam. While we observe similar behavior it isn't as pronounced as we'd expect given the large change in novel samples from WWU Webcam to ADE20K. We offer the hypothesis that this is because WWU Webcam is so repetitive that even Random ordering has a repetitive and Sequential nature to it.

Overall different Data Ordering strategies have their place. Both Sequential and Tortoise-Hare seem to be poor choices. A Random N-Chunk can perform quite well, with low N numbers performing better than expected on all tasks but classification. This indicates it is a viable strategy to sequentially read data into a more efficient storage location (such as VRAM) before randomizing it their.

Both Random and SBMFRAW perform well on all tasks. Though Random seems to perform better on all tasks except WWU Webcam Reconstruction. This shows there is viability in semi-structured random ordering strategies, though SBMFRAW might not be the precise implementation to maximize variety in abnormal datasets.

Finally we note the lack of difference between ResNet18 and ResNet50, suggesting to us that the Data Ordering doesn't change too heavily depending on the depth of Residual Networks at least.

B. Limitations

Due to a lack of compute resources many of these experiments were run only once. It is possible that on many iterations further or less obvious patterns would emerge. Additionally, results observed could be due to random noise, that would be filtered out over many iterations.

This also limited the studying of Residual Networks. ResNet18 and 50 are trained for many more epochs than the 10 used in this research, and new trends could appear as we get closer to convergence in classification.

All models trained use convolutional architectures and supervised training; alternatives such as contrastive methods or attention-based architectures may show different results

C. Future Work

Our findings show viability in algorithmic Data Ordering strategies. Research into Curriculums may be able to apply the knowledge discovered about data variety to their easy-to-hard Data Ordering strategies to maximize benefits.

More experiments on different models and datasets would seek to deepen and broaden our understanding. As mentioned before non-convolutional architectures could value a very different type of variety than convolutional architectures. Similarly datasets with different structures could prove insightful. What does variety look like for human faces, or astrological photos.

Other Data Ordering strategies could also be employed. Is there some way to quantify the variety we are observing, and order in a way which maximizes it. We'd also like to know if

there is a viable non-random Data Ordering strategy. Tortoise-Hare fails to be competitive with random methods, but we would like to see a method that could efficiently sample videos given their encoding

VI. Conclusion

In this paper we presented scenarios where different Data Ordering strategies were viable, allowing more efficient strategies to be chosen based on your specific dataset. Additionally, we presented a novel Data Ordering strategy SBMFRAW, which showed that in some scenarios there are strategies better than random.

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[2] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples, 2018.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[4] Qi Meng, Wei Chen, Yue Wang, Zhi-Ming Ma, and Tie-Yan Liu. Convergence analysis of distributed stochastic gradient descent with shuffling, 2017.

[5] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[6] Daphna Weinshall and Gad Cohen. Curriculum learning by transfer learning: Theory and experiments with deep networks. *CoRR*, abs/1802.03796, 2018.

[7] Weihao Xia, Zhanglin Cheng, Yujiu Yang, and Jing-Hao Xue. Cooperative semantic segmentation and image restoration in adverse environmental conditions, 2020.