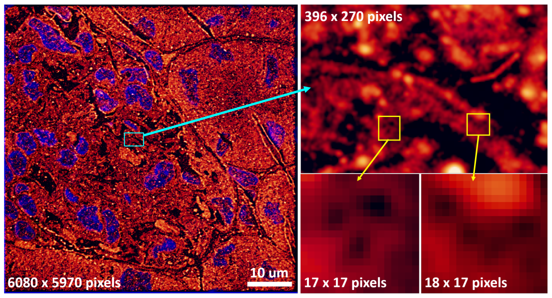

Figure 3. Example nanoscopy image (left) of a mouse kidney cryosection approximately 1/12th of the area of a single field-of-view of the microscope, chosen to illustrate the level of details at different scales. The bottom right images show that the smallest features in the image of relevance can be as small as a few pixels (here 5-8 pixels for the holes)[?].

A. Broader Impact

The broader impact of this work lies particularly in its potential to extend the capability of deep learning models. By addressing the challenge of training models on large-scale images with limited computational resources, our approach opens up opportunities for researchers and practitioners with constrained hardware setups to tackle complex problems in healthcare, agriculture, and environmental monitoring, where high-resolution images play a crucial role in decision-making processes. Moreover, our approach can contribute to reducing the environmental footprint of deep learning by enabling efficient training on low-power devices, thus promoting sustainability in the development and deployment of deep learning models. In summary, our work has the potential to empower diverse communities, drive sustainable development, and accelerate scientific progress. It is essential to approach these advancements with a conscientious mindset, taking into account the broader societal impact and proactively working towards an inclusive and responsible deployment of deep learning technologies. With our work, it is also important to address the potential risks and challenges. Issues related to data privacy, bias, and fairness should be carefully addressed to prevent any unintended negative consequences. Additionally, the potential for misuse or malicious applications of deep learning models should be acknowledged and proactively addressed through robust security measures and ethical guidelines.

B. Future work

This paper has established the foundational concept of patch gradient descent to enable training CNNs using very large images and even when only limited GPU memory is available for training. The results as well as insights presented in the paper open doors to several novel secondary research directions that could be interesting in terms of improving the efficacy as well as the acceptance of the presented method in a broader scientific community. We list some such directions here.

- *Scaling to gigapixel images at small compute memory.* An ambitious but very interesting application of PatchGD would be to be able to process gigapixel images with small GPU memory. We can clearly envision this with PatchGD but with additional work. One important development needed is to extend the PatchGD learning concept to multiple hierarchical **Z** blocks, thereby sampling patches from the outer block to iteratively fill the information in the immediate inner Z block and so on.
- *Enhanced receptive field.* So far, PatchGD has been looked at only in the context of being able to handle very large images. However, a different side of its use is that with almost the same architecture, it builds a smaller receptive build, thereby zooming in better. We speculate that in this context, PatchGD could also help in building better discriminative models with lighter CNN architectures. Clearly, this would be of interest to the deep learning community and needs to be explored.
- *PatchGD with Transformers.* Transformers are known to provide a better global context and it would be interesting to expand the capability of transformers as well to handle large images using PatchGD.

C. Datasets

C.1. PANDA

The Prostate cANcer graDe Assessment Challenge [1] consists of one of the largest publically available datasets for Histopathological images which scale to a very high resolution. It is important to mention that we do not make use of any masks as in other aforementioned approaches. Therefore, the complete task boils down to taking an input high-resolution image and then classifying them into 6 categories based on the International Society of Urological Pathology (ISUP) grade groups. There are a total of 10.6K images which are split into train and test sets in the ratio 80:20.

C.2. UltraMNIST

This is a synthetic dataset generated by making use of the MNIST digits. For constructing an image, 3-5 digits are sampled such that the total sum of digits is less than 10. Thus an image can be assigned a label corresponding to the sum of the digits contained in the image. Each of the 10 classes from 0-9 has 1000 samples making the dataset sufficiently large. Note that the variation used in this dataset is an adapted version of the original data presented in [3], with background noise removed so that any shortcut learning is avoided [2]. Since the digits vary significantly in size and are placed far from each other, this dataset fits well in terms of learning semantic coherence in a image. Moreover, it poses the challenge that downscaling the images leads to a significant loss of information. While even higher resolution could be chosen, we later demonstrate that the chosen image size is sufficient to demonstrate the superiority of PatchGD over the conventional gradient descent method.

C.3. TCGA

The TCGA-NSCLC dataset, known as The Cancer Genome Atlas-Non-Small Cell Lung Cancer, encompasses two distinct types of lung cancer: Lung Adenocarcinoma (LUAD), with 522 cases, and Lung Squamous Cell Carcinoma (LUSC), with 504 cases, with a total number of image files 3220. The data was split in a stratified manner using the patient cases, into train and test set in the ratio 80:20, making sure there is no data leakage from train to test. The whole slide images are used to evaluate the performance of baseline and PatchGD in classifying the lung cancer subtypes.

C.4. ImageNet100

The ImageNet100 is a derived dataset from the parent ImageNet[4]. The dataset consists of 100 randomly chosen classes from the original 1000, which are used to perform classification via both the baselines as well as PatchGD.

D. Training Methodology and Hyperparameters

For Tables 1,2,3,5,6,7 presented in the main paper, all models are trained for 100 epochs with Adam optimizer and a peak learning rate of 1e-3. A learning rate warm-up for 2 epochs starting from 0 and linear decay for 98 epochs till half the peak learning rate was employed. The latent classification head consists of 4 convolutional layers with 256 channels in each. We perform gradient accumulation over inner iterations for better convergence, in the case of PANDA. To verify if results are better, not because of an increase in parameters (coming from the classification head), baselines are also extended with a similar head. GD*, for MobileNetV2 on UltraMNIST, refers to the baseline extended with this head.

In the case of low memory, as demonstrated in the UltraMNIST experiments, the original backbone architecture is trained separately for 100 epochs. This provides a better initialization for the backbone and is further used in PatchGD as mentioned in Tables 1 and 2.

For baseline in PANDA at 2048 resolution, another study involved gradient accumulation over images, which was done for the same number of images that can be fed when the percent sampling is 10% i.e. 14 times since a 2048x2048 image with a patch size of 128 and percentage sampling of 10 percent can have a maximum batch size of 14 under 16GB memory constraint. That is to say, the baseline can virtually process a batch of 14 images. This, however, was not optimal and the peak accuracy reported was in the initial epochs due to the loading of the pre-trained model on the lower resolution after which the metrics remained stagnant(accuracy: 32.11%, QWK:0.357).

For Table 4 presented in the main paper, we use the respective training strategies as mentioned in the respective works. The training strategy on TCGA is similar to what is employed on the PANDA dataset. In the case of ImageNet100, both the baselines and PatchGD were trained with a peak learning rate of 1e-3, with Adam Optimizer. Cosine LR decay with warmup was used as the learning rate scheduler. The image level augmentation pipelines are implemented as in the A3 pipeline of [6] both for baseline and PatchGD. For PatchGD, the stride(20) is kept to be half of the patch size(40) and the percent sampling was 25%. under 3GB memory constraint.

E. On other tasks

Generative modeling. PatchGD can be used for generating large-scale images with a broad semantic context, which can be beneficial for data augmentation in fields such as deep learning for medical imaging. Early results using StyleGAN-2 on the CIFAR-10 dataset showed that our method generated patches of 16×16 which were stitched together and analyzed by the discriminator, leading to a comparable FID score of 6.3 to the standard GD's FID score of 6.1. We believe this small performance gap can be eliminated with hyperparameter optimization. We consider that the potential of PatchGD in generative modeling can be maximized by generating large images with various semantic contexts, although this needs to be explored further.

PatchGD for segmentation. We discuss here how PatchGD can be used for tasks such as segmentation or any other encoder-decoder tasks We have discussed generative modeling already, and since the setup would be something similar, we present here an understanding of how the PatchGD formulation would unfold for tasks such as segmentation. For the task of segmentation as well, we have two sets of weights θ_1 and θ_2 that constitute the encoder and the decoder, respectively. Here, the encoder generates a Z-block and the decoder is used to generate the segmentation map from the Z-block. Similar to the classification problem, PatchGD operates on each image over a course of multiple inner iterations. At each inner iteration, patches are sampled from image x and accordingly passed through and the output is then used to update the respective parts of Z. Further, k c-dimensional vectors are sampled from Z and passed through the decoder to generate mask patches that are used to update parts of the segmentation map y, and the process is repeated. Note that similar to Z–filling, this process also requires y-filling before the model updates of the encoder and decoder are performed over patches. For

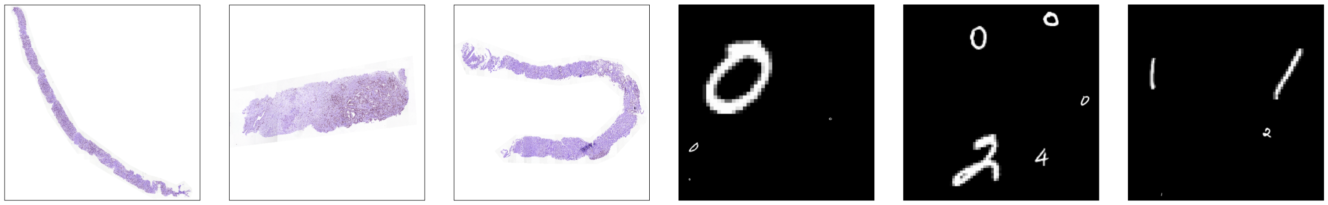
Figure 4. Sample PANDA and UltraMNIST dataset images used for training PatchGD.

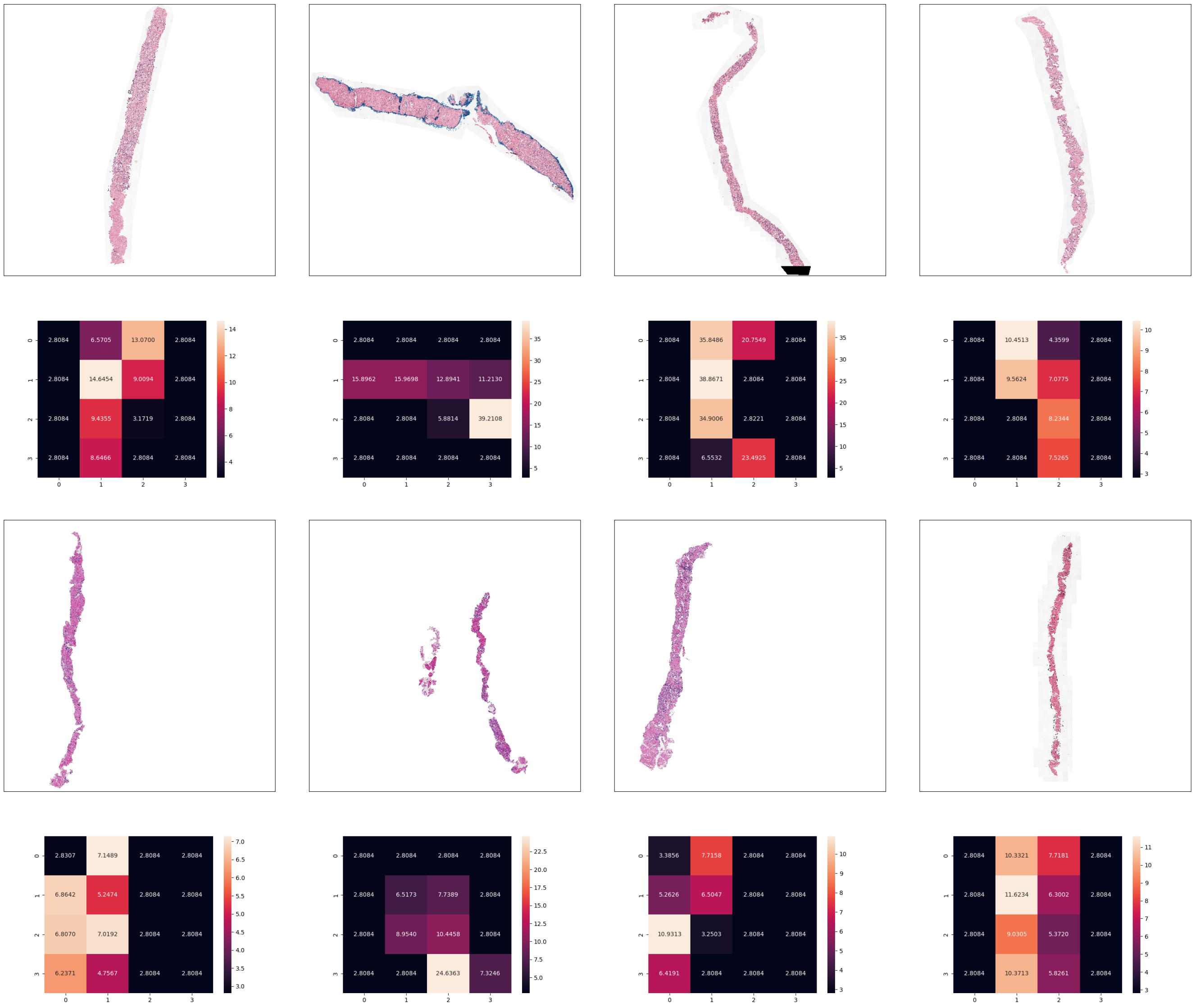
Figure 5. Sample PANDA images along with their latent space Z. It can be seen that the latent space clearly acts as a rich feature extractor.

this purpose, we can first train a segmentation model on lower-resolution images of the chosen task and then use its encoder and decoder, and starting models for the PatchGD learning process.

F. Comparison with normalization techniques

Batch normalization methods also influence the covergence of deep learning models at low batch sizes. However, PatchGD outperforms these techniques as well and we present a comparison is presented in Table 1.

Table 1. Comparison with normalization techniques at 2048 image size and 48GB memory constraint with Resnet50 backbone.

Method	Batch Size	Setting	Accuracy %
BatchNorm	6	-	49.4
GroupNorm	6	Groups = 32	50.3
Grad. Acc.	5	Steps = 11	44.1
PatchGD	56		56.2

G. Gradient Accumulation Study

We also highlight an ablation study on the effect of changing the gradient accumulation steps ϵ as presented in Table 2. The gradients are accumulated and weights are updated only after ϵ steps. The ablations were conducted for different epsilon settings, image and patch sizes, and memory constraints. We found that for smaller patch sizes, employing gradient accumulation steps greater than 1 is essential, with significant gains observed as the patch size to image size ratio decreases. Despite this promising trend,ϵ remains a hyperparameter requiring further tuning. Moreover, exploring the nuanced relationship between accuracy and steps is an essential aspect for future investigation in optimizing PatchGD. In case of UltraMNIST dataset at 512 image size, best performance is observed at $\epsilon = 1$ for a patch size of 256. For PANDA two variations were tried for image size 512 and image size 4096 with best results obtained at 8 and 32 respectively.

H. Applications in Time Series Classification

Extending the concept of PatchGD to the 1-dimensional case, we find the application in time series classification. For this task, we take the example of UCI Human Activity Recognition Dataset [5]. A set of 9 inertial signals at 128 unique time stamps are used to predict the action being executed (sitting, walking, etc.). For the baseline model, we use a basic 1-d Convolutional Network with 64 kernels each of size 3 and a linear layer at the end which achieves an accuracy of 88.9%. The model is trained using Adam as an optimizer with a constant learning rate of 1e-3 for 30 epochs with 32 batch size. The counterpart PatchGD-inspired approach involved the same 1-d convolutional network as the encoder with an intermediate latent vector, with other common hyperparameters being kept the same. The time series is broken into chunks temporally, each chunk being of length 16. Each inner iteration consists of sampling 25% of the total chunks with gradient updates enabled. The model is updated at the final iteration. Impressively, the approach achieves similar accuracy of 88.5%. The results are promising and yet again demonstrate the wide application to other tasks where PatchGD can be applied. Although this needs to be investigated further.

References

[1] Wouter Bulten, Kimmo Kartasalo, Po-Hsuan Cameron Chen, Peter Ström, Hans Pinckaers, Kunal Nagpal, Yuannan Cai, David F Steiner, Hester van Boven, Robert Vink, et al. Artificial intelligence for diagnosis and gleason grading of prostate cancer: the panda challenge. *Nature medicine*, 28(1):154–163, 2022. 1

[2] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020. 1

[3] Deepak K. Gupta, Udbhav Bamba, Abhishek Thakur, Akash Gupta, Suraj Sharan, Ertugrul Demir, and Dilip K. Prasad. Ultramnist classification: A benchmark to train cnns for very large images. *arXiv*, 2022. 1

[4] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015. 2

[5] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM conference on embedded networked sensor systems*, pages 127–140, 2015. 4

[6] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 2

Table 2. Influence of different number of gradient accumulation steps ϵ on the performance of PatchGD.

Model	Dataset	Memory	Image size	Patch size	ϵ	Accuracy
MobileNetv2	UltraMNIST	16 GB	512	256	1	**83.7**
MobileNetv2	UltraMNIST	16 GB	512	256	2	81.5
MobileNetv2	UltraMNIST	16 GB	512	256	4	81.1
Resnet50	PANDA	4GB	512	64	1	41.9
Resnet50	PANDA	4GB	512	64	8	**50.5**
Resnet50	PANDA	4GB	512	64	32	45.0
Resnet50	PANDA	48GB	4096	256	8	56.9
Resnet50	PANDA	48GB	4096	256	32	**59.7**