Streaming Convolutional Neural Networks for End-to-End Learning With Multi-Megapixel Images

Hans Pinckaers, Bram van Ginneken, and Geert Litjens

Abstract—Due to memory constraints on current hardware, most convolution neural networks (CNN) are trained on sub-megapixel images. For example, most popular datasets in computer vision contain images much less than a megapixel in size (0.09MP for ImageNet and 0.001MP for CIFAR-10). In some domains such as medical imaging, multi-megapixel images are needed to identify the presence of disease accurately. We propose a novel method to directly train convolutional neural networks using any input image size end-to-end. This method exploits the locality of most operations in modern convolutional neural networks by performing the forward and backward pass on smaller tiles of the image. In this work, we show a proof of concept using images of up to 66-megapixels (8192×8192), saving approximately 50GB of memory per image. Using two public challenge datasets, we demonstrate that CNNs can learn to extract relevant information from these large images and benefit from increasing resolution. We improved the area under the receiver-operating characteristic curve from 0.580 (4MP) to 0.706 (66MP) for metastasis detection in breast cancer (CAMELYON17). We also obtained a Spearman correlation metric approaching state-of-the-art performance on the TUPAC16 dataset, from 0.485 (1MP) to 0.570 (16MP). Code to reproduce a subset of the experiments is available at https://github.com/DIAGNijmegen/StreamingCNN.

Index Terms—Deep learning, convolutional neural networks, image classification, high-resolution images

1 Introduction

Convolutional neural networks (CNN) are the current state-of-the-art machine learning algorithms for many computer vision tasks, such as classification or segmentation. Ever since Krizhevsky *et al.* won ImageNet [1] with a CNN [2] in 2012, these networks have become deeper [3] and wider [4] to further improve accuracy. Training these larger networks requires large amounts of computer memory, which increases exponentially with increasing image size. To avoid shortcomings in memory, most natural image datasets in computer vision contain sub-megapixel images: 0.09 megapixel for ImageNet [1] and 0.001 megapixel for CIFAR-10 [5]. In several domains such as remote sensing or medical imaging, there is a need for training CNNs with multi-megapixel-sized images – containing both global contextual and local textural information – to obtain accurate models.

Computer memory becomes a limiting factor because the conventional backpropagation algorithm for optimizing deep neural networks requires the storage of intermediate activations. Since the size of these intermediate activations in a convolutional neural network increases proportionally to the input size of the network, memory quickly fills up with images of multiple megapixels. As such, only small CNNs could be trained with these images and state-of-the-art architectures would be out of reach, even on large computing clusters.

In this paper, we propose a novel method to directly train state-of-the-art convolutional neural networks using any input image size end-to-end. This method exploits the locality of most operations in modern convolutional neural networks by tiling the forward and backward pass in combination with gradient checkpointing. Gradient checkpointing is a technique where instead of keeping all intermediate feature maps in memory to calculate the gradients, we save some specific feature maps (checkpoints). We recalculate the others by performing partial forward passes starting from the saved feature maps during backpropagation, once they are needed for gradient calculation [6]. We first empirically established equivalence between our tile-based approach and an unmodified convolutional neural network on a subset of ImageNet, ImageNette [7]. Then we applied this method to two public datasets: the CAMELYON17 dataset [8] for metastases detection in lymph nodes, and the TUPAC16 dataset [9] for predicting a proliferation score based on gene expression. In both cases, task-specific performance increased with larger input image sizes.

2 Related Work

Several authors have suggested approaches to train convolutional neural networks (CNNs) with large input images while preventing memory bottlenecks. Their methods can be roughly grouped into three categories: (A) altering the dataset, (B) altering usage of the dataset, and (C) altering the network or underlying implementations.

• *The authors are with the Diagnostic Image Analysis Group, Radboud Institute for Health Sciences, Radboud University Medical Center, 6525, GA, Nijmegen, The Netherlands. E-mail: {hans.pinckaers, bram.vanginneken, geert.litjens}@radboudumc.nl.*

Manuscript received 11 Nov. 2019; revised 26 July 2020; accepted 20 Aug. 2020. Date of publication 26 Aug. 2020; date of current version 3 Feb. 2022.
(Corresponding author: Hans Pinckaers.)
Recommended for acceptance by T. Arbel.
Digital Object Identifier no. 10.1109/TPAMI.2020.3019563

0162-8828 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF BIRMINGHAM. Downloaded on April 18,2024 at 10:13:11 UTC from IEEE Xplore. Restrictions apply.

2.1 Altering the Dataset

If images are too large to fit in the memory of the processing unit, we could downsample the image or divide the image into smaller parts, i.e., patches. The latter approach has been prevalent in both remote sensing and medical imaging [10], [11]. However, both approaches have significant drawbacks: the former results in a loss of local details, whereas the latter results in losing global contextual information.

The common approach of training on patches typically involves creating labels for every patch, which can be time- and cost-intensive. It is sometimes not even possible to produce patch-level labels: if a hypothetical task is to predict whether an aerial image shows a city or a village, it is impossible to create informative labels for individual patches only containing several houses.

2.2 Altering Usage of the Dataset

When we can assume that individual patches contain enough information to predict the image-level label, the classification can be formalized under the classic multiple-instance-learning (MIL) paradigm. In MIL, each image is considered a bag consisting of patches where a positive bag has at least one positive patch and a negative bag none. In the deep learning case, a model is trained in a weakly supervised manner on patches, where the patch with the highest predicted probability is used for backpropagation [12], [13]. Other approaches involve taking the average of the patch predictions or a learned weighted average from low-dimensional patch embeddings [14], [15], [16], [17], [18].

In this approach, the receptive field of a network is always at most the size of the patch. The model disregards spatial relationships between patches, limiting the incorporation of contextual information.

By first learning to decide which regions should be analyzed at a higher resolution, the problem that a full image cannot be used can also be circumvented [19], [20], [21], [22]. Since these methods use a low-resolution version of the image to decide which parts need to be analyzed at a higher resolution, the low-resolution image needs to have enough information to localize the area that needs to be classified. Additionally, for analysis of the selected areas, these methods still use patch-based analysis with the same caveats as mentioned before.

Another way to utilize datasets with large images is proposed by Tellez *et al*. [23]. To compress the image to a lower-dimensional space, they proposed unsupervised learning. The model is trained patch-by-patch to reconstruct the original patch. An intermediate feature map of the model (i.e., the embedding) can subsequently be used as a lower-dimensional representation per patch. After training, the whole image is compressed patch-by-patch. A model is subsequently trained on these embeddings, having the receptive field of the whole image while requiring less memory.

Since the compression network is trained by reconstruction, the same compression network can be used for different tasks. However, this means that the low-dimensional embedding is not meant for a specific task and may have compressed away useful information. Our approach involves one network which learns to compress task-relevant information.

2.3 Altering the Network or Underlying Implementations

The memory bottleneck can also be circumvented with memory-efficient architectures or memory-efficient implementations of existing architectures. Recently, Gomez *et al*. [24] published a method to train deep residual neural networks using less memory, termed the Reversible Residual Network. With these networks, some layer activations are recomputed from others on demand, reducing the total memory required. Network architectures can also be altered to utilize cheaper computational operation, such as depth-wise separable convolutions [25] or fewer parameters [26]. Our method does not require reducing the number of parameters and works with most types of layers. Another method to reduce memory usage is to recover intermediate activations by doing partial forward passes during backpropagation, termed gradient checkpointing [6]. This method is similar to our approach, but the whole activation feature map of some layers still need to be stored in memory, limiting the use of multi-megapixel images.

Another memory-saving approach is to share memory between tensors with duplicate or recomputable values [27], [28], to develop neural networks with reduced precision using half-precision or mixed precision [29], or to swap data between random access memory (RAM) and graphics processing unit (GPU) memory [30]. These methods are usually insufficient for training with large multi-megapixel images; our proposed method can work orthogonally to them.

3 Methods

To achieve our goal of training CNNs with multi-megapixel images, we significantly reduce the memory requirements. Memory demand is typically highest in the first few layers of state-of-the-art CNNs before several pooling layers are applied because the intermediate activation maps are large. These activation maps require much less memory in subsequent layers. We propose to construct these later activations by streaming the input image through the CNN in a tiled fashion, changing the memory requirement of the CNN to be based on the size of the tile and not the input image. This method allows the processing of input images of any size.

Several problems arise when trying to reconstruct the later activation map tile-by-tile. First, convolutional layers handle image borders in different ways, either by padding zeros to perform a "same" convolution or by reducing the image size to perform a "valid" convolution. Second, in tile-based processing, border effects occur at both the image borders and the tile borders; naive tiling of the input image would thus result in incomplete activation maps and gradients for backpropagation. Lastly, intermediate feature maps of the tiles still need to be stored in memory for backpropagation, which would counteract the streaming of tiles. We solve these problems by developing a principled method to calculate the required tile overlap throughout the network in both the forward and backward pass and by using gradient checkpointing.

We first explain the reconstruction of the intermediate activation map in the forward pass in Section 3.1, then describe the backward pass in Section 3.2, elaborate on how to calculate the tile overlap in Section 3.3, and finish with

Authorized licensed use limited to: UNIVERSITY OF BIRMINGHAM. Downloaded on April 18,2024 at 10:13:11 UTC from IEEE Xplore. Restrictions apply.

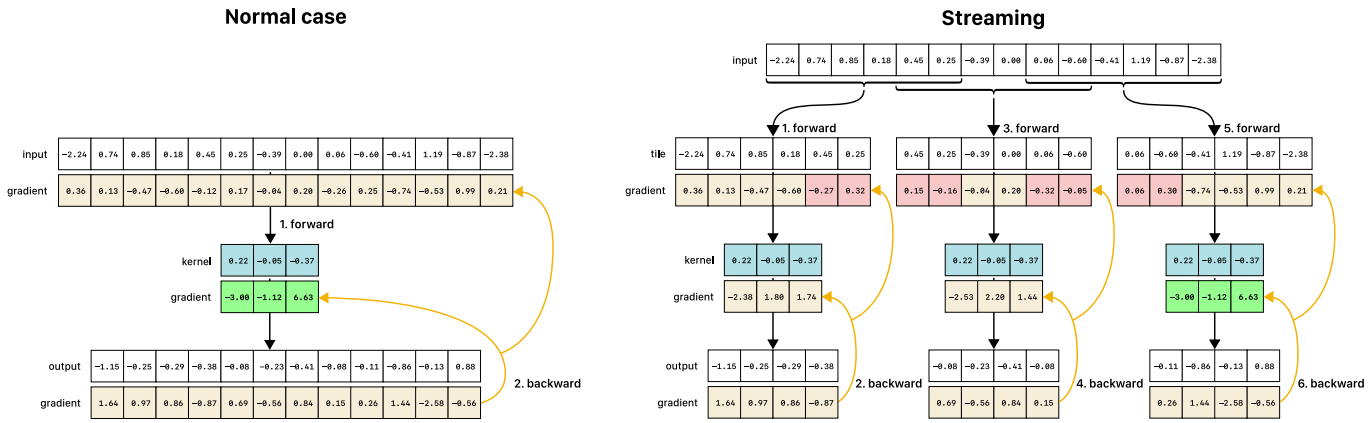

Fig. 1. Schematic overview of streaming in a one-dimensional case, using a 1×3 kernel. We can calculate the same gradients with smaller input and output shapes, saving memory. Colored in red are incomplete gradients of the input tile, illustrating that if we chain two of these one-dimensional convolutions together, more overlap is needed (four values instead of the two pictured). During streaming, the gradients of the convolutional kernel are not reset between tiles and are summed with the previous gradients. After the three tiles have been backpropagated, the kernel gradients are equal to the normal case. Gradient checkpointing is also used to save additional memory, but is omitted here for clarity. Numbers denote the order of serial steps taken by the respective algorithms.

the limitations of this method in Section 3.4. See Fig. 1 for a graphical representation of the method.

3.1 Streaming During the Forward Pass

Without loss of generality, we explain the method in the discrete one-dimensional case. Let us define $x \in \mathbb{R}^N$ as the one-dimensional real-valued vector with N elements. In discrete one-dimensional space, a "valid" convolution[1] (*) with a kernel with n weights $w \in \mathbb{R}^n$, and stride 1, is defined as

$$(x * w)_k = \sum_{i=0}^{n} w_i x_{k+i}, \tag{1}$$

where $k \in \{0, \ldots, f\}$ and $f = N - n$, for any kernel with length $n \leq N$ (for clarity, we will start all indices from 0). Our goal is to decrease the memory load of an individual convolution by tiling the input. Following (1), we can achieve the same result as $x * w$, by doing two convolutions on the input

$$a = \{(x * w)_0, \ldots, (x * w)_{f//2}\} \tag{2}$$

$$b = \{(x * w)_{f//2+1}, \ldots, (x * w)_f\}, \tag{3}$$

where // denotes a divide and floor operation.

By definition of concatenation ($\frown$)

$$\{(x * w)_0, \ldots, (x * w)_f\} = a \frown b. \tag{4}$$

To ensure that the concatenation of both tiles results in the same output as for the full vector, we need to increase the size of the tiles, resulting $o = n - 1$ overlapping values. The values $\{x_0, \ldots, x_{f//2+o}\}$ are required to calculate a, and $\{x_{f//2+1-o}, \ldots, x_N\}$ for b.

Since the tiles are smaller than the original vector, these separate convolutions require less memory when executed in series. By increasing the number of tiles, memory requirements for individual convolution operations can be reduced even further.

Without loss of generality, the above can also be extended to multiple layers in succession including layers with stride > 1 (e.g., strided convolutions and pooling layers) which are also commonly used in state-of-the-art networks.

When one applies this tiling strategy naively, no memory benefit is obtained as each tile's intermediate activation would still be stored in memory to allow for backpropagation. We use gradient checkpointing to resolve this: We only store the activations after the concatenation of the tiles – where the memory burden is small. This does require recalculation of all intermediate activations for all tiles during backpropagation, but again, only has a memory requirement of processing of a single tile. The trade-off between memory use and re-computation can be controlled through the selection of the concatenation point in the network.

From this point onward, the term *streaming* refers to the tiling of a vector, applying kernel operations, and concatenating the results.

3.2 Streaming During Backpropagation

The backward pass of multiple convolutions can also be calculated by utilizing the tiles. To start, let us define p as the output after streaming. The derivative of a weight in a convolutional kernel is defined as

$$\Delta w_j = \sum_{i=0}^{|p|-1} \begin{cases} \Delta p_i x_{i+j}, & \text{if } i - j \geq 0 \text{ and } i - j < |p| \\ 0, & \text{otherwise} \end{cases}, \tag{5}$$

where $|\cdot|$ denotes the length of a vector.

While streaming, this sum has to be reconstructed through the summation of the gradients of all tiles, which will result in the same gradient again

$$\Delta w_j = \sum_{i=0}^{|a|-1} \Delta a_i x_{i+j} + \sum_{i=0}^{|b|-1} \Delta b_i x_{i+j+f//2}. \tag{6}$$

The gradient of the input can be calculated with a similar sum, but then shifted by the kernel size

$$\Delta x_i = \sum_{j=0}^{n-1} \begin{cases} w_j \Delta p_{i-j}, & \text{if } i - j \geq 0 \text{ and } i - j < |p| \\ 0, & \text{otherwise} \end{cases}. \tag{7}$$

1. By convention we used the term *convolution* although the mathematical operation implemented in most machine learning frameworks (e.g., TensorFlow, PyTorch) is a cross-correlation.

Authorized licensed use limited to: UNIVERSITY OF BIRMINGHAM. Downloaded on April 18,2024 at 10:13:11 UTC from IEEE Xplore. Restrictions apply.

This formula is equal to a convolution with a flipped kernel w on Δp padded with $n-1$ zeros (e.g., $flip(w) * [0, 0, \Delta p_1 \dots \Delta p_n, 0, 0]$, when $n = 3$), often called a "full" convolution. Thus, analog to the forward pass, the backpropagation can also be streamed.

However, overlapping values of the output p are required when streaming the backpropagation, similar to the overlapping values of the input x required in the forward pass. To generate overlapping values for the output p, the overlap o for the input x needs to be increased to calculate the full Δx.[2]

Algorithm 1. Forward and Backward Pass With Streaming Through the Bottom Layers of the Network With Tiles

in :n convolutional $layers$, x image to stream with m tiles t at $coordinates$, and i the last layer to stream. `crop_unique` uses indices from *Algorithm 2*.
out :$grads$ containing gradients per layer.

$o \leftarrow []$ ▷ array to collect tile outputs
with *no gradient computation*
  for c *in coordinates* **do**
    $t \leftarrow \texttt{crop}(x, c)$
    $o[c] \leftarrow \texttt{forward}(layers[0..i], t)$
  end
end
$stream\_o \leftarrow \texttt{concat}(o[0..m])$
$pred \leftarrow \texttt{forward}(layers[i..n], stream\_o)$
$loss \leftarrow \texttt{criterion}(pred)$
$g \leftarrow \texttt{backward}(layers[n..i], loss)$
$filled \leftarrow []$ ▷ array to remember backpropped indices
for c *in coordinates* **do**
  $o[c] \leftarrow \texttt{forward}(layers[0..i], t)$ ▷ checkpointing
  $g_t \leftarrow \texttt{crop\_relevant\_gradient}(g, c)$
  for l *in* $layers[i..0]$ **do**
    $g_t \leftarrow \texttt{backward}(l, g_t)$
    $o_u, g_u, filled[l] \leftarrow \texttt{crop\_unique}(l, o[c], c, filled[l])$
    $g_{kernel} \leftarrow \texttt{backward}(o_u, g_u)$
    $grads[l] \leftarrow \texttt{sum\_gradients}(g_{kernel}, grads[l])$
  end
end

3.3 Efficiently Calculating Required Tile Overlap for Complex Architectures

Some recent state-of-the-art networks (e.g., ResNet and DenseNet) contain different paths through the network that sum or concatenate activations from different layers together. These paths make it difficult to manually calculate the required tile overlap for streaming.

To calculate the overlap for such networks, we temporarily replace all convolutional kernel parameters with $\frac{1}{n}$, where n was the length of the kernel. This causes each entry in the convolutional layer's output to be the average of the input image spanned by the convolutional kernel. We then pass an all-ones tile through the network. The required overlap will be the number of non-maximum values in the activation maps and gradients at the border of the tiles, see Algorithm 2.

2. Zero-padding the tiles before the convolution does not help because these zeros do not exist in the original vector, hereby invalidating the gradients at the border as well.

Algorithm 2. Finding Which Areas of Streamed Feature Maps and Gradients Contain Values Equal to Feature Maps and Gradients in an Backpropagated Full-Resolution Image

in: t tile at the desired tile size, containing a constant value, $layers$ containing n number of convolutional neural network layers, and i the last layer to stream.
out: statistics for forward and backward pass $invalid\_forw\_i$, $invalid\_back\_i$, and $output\_stride$.

$output\_stride \leftarrow 1$
for l *in* $layers[0..i]$ **do**
  $kernel\_backup_l \leftarrow kernel_l$
  $kernel_l \leftarrow 1/\texttt{len}(kernel_l)$
  $t \leftarrow \texttt{forward}(l, t)$
  $invalid\_forw\_i \leftarrow \texttt{non\_max\_indices}(o)$
  $output\_stride \leftarrow output\_stride * l.stride$
end
$o \leftarrow \texttt{forward}(layers[i..n], o)$
$loss \leftarrow \texttt{criterion}(o)$
$g \leftarrow \texttt{backward}(layers[n..i], loss)$
for l *in* $layers[i..0]$ **do**
  $g \leftarrow \texttt{backward}(l, g)$
  $invalid\_back\_i \leftarrow \texttt{non\_max\_indices}(g)$
  $kernel_l \leftarrow kernel\_backup_l$
end

3.4 Limitations

With small tiles, the overlap can be a significant part of the tile, counteracting the memory gains. Since we leverage the method for high-resolution images using large tiles, the memory gains outweigh this overhead.

Furthermore, due to the use of gradient checkpointing, the method will perform multiple forward and backward operations to calculate intermediate activations. This results in longer processing time than it would take if the image could fit on the GPU (see Figs. 2 and 3). The processing time increases almost linearly with input size plus overlap.

For a network to be able to use this method, the intermediate feature maps and its gradients have to be able to fit on the GPU at a certain point. However, choosing a layer too deep into the network will require a lot of overlapping calculations, being less efficient. As such, choosing which layers to stream can be difficult. We suggest splitting the network and experimenting with random input to the final non-streaming layers to test if backpropagation fits on the GPU. Then, streaming the first layers with a tile size as large as possible.

Finally, since the method relies on the local properties of convolutions and pooling operations, trying to use other operations that break this locality will result in invalid results (e.g., operations that rely on all the feature map values such as BatchNormalization [31]). However, these operations can be used as soon as the whole feature map is reconstructed, after streaming, in the final part of the network.

4 Evaluation

We evaluated the streaming method with three different datasets and network architectures. First, in Section 5, we

Authorized licensed use limited to: UNIVERSITY OF BIRMINGHAM. Downloaded on April 18,2024 at 10:13:11 UTC from IEEE Xplore. Restrictions apply.

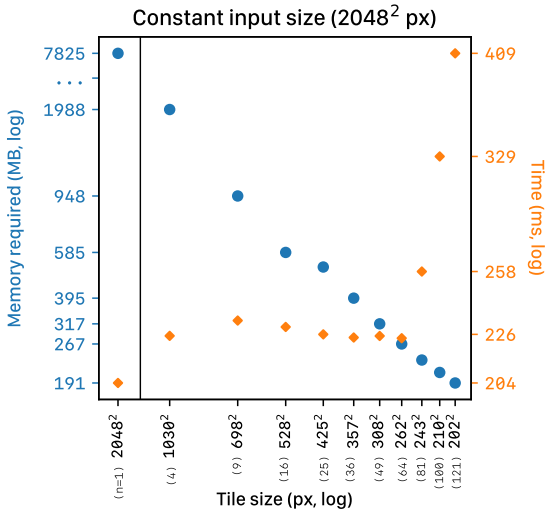

Fig. 2. This graph shows the linear relationship between the size of the tile and the memory required to stream an input of size 2048×2048. Since the amount of overlap is minimal in this example (16 pixels), the time does not increase until 81 tiles are needed. Performance shown of three 2D-convolutional layers with respectively 3, 64, and 3 output channels and kernel-size 3 on an RTX 2080Ti GPU. All numbers are averaged over 10 runs.

evaluated whether a CNN using streaming trains equivalently to the conventional training. Second, in Section 6, we evaluated the usage of streaming on a regression task in the public TUPAC16 [9] dataset with high-resolution images (multiple gigapixels) and only image-level labels. We trained multiple networks using increasing image resolutions and network depth. Finally, in Section 7, we evaluated streaming in a classification task using the image-level labels of the CAMELYON17 dataset [32].

An open-source implementation of the streaming algorithm and the ImageNette experiments can be found at https://github.com/DIAGNijmegen/StreamingCNN.

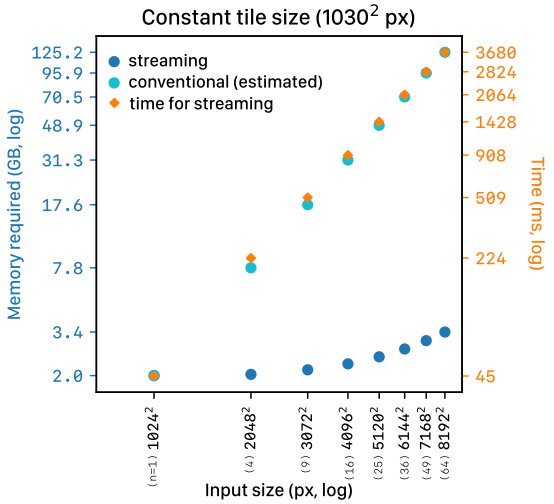

Fig. 3. This graph shows the linear relationship with time required when increasing the input size (keeping the tile size constant). With streaming, there is significantly less memory required (up to 97 percent). The memory increase with streaming is due to the size of the input and output gradient. The amount of memory required to train with images above 2048 × 2048 without streaming is estimated. Performance shown of three 2D-convolutional layers with respectively 3, 64, and 3 output channels and kernel-size 3 on an RTX 2080Ti GPU. All numbers are averaged over 10 runs.

TABLE 1
Network Architecture for Imagenette Experiment

Layers	Kernel size	Channels
2D convolution	3x3	16
2D max-pool	2x2	16
2D convolution	3x3	32
2D max-pool	2x2	32
2D convolution	3x3	64
2D max-pool	2x2	64
2D convolution	3x3	128
2D max-pool	2x2	128
2D convolution	3x3	256
2D max-pool	10x10	256
Fully connected	10	

5 Experiments on ImageNette

We trained a CNN on small images using streaming and conventional training starting from the same initialization. We used a subset of the ImageNet dataset, ImageNette, using 10 ImageNet classes (tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, parachute), analog to [7].

5.1 Data Preparation

We used data augmentation for the training set following Szegedy *et al.* [33]. Patches of varying sizes were sampled from the image, distributed evenly between 8 and 100 percent of the image area with aspect ratio constrained to the interval $[\frac{3}{4}, \frac{4}{3}]$. For the tuning set, we sampled 320 × 320 patches from the center of the image.

5.2 Network Architecture and Training Scheme

The CNN consisted of five blocks of a convolutional layer followed by a max-pool layer (see Table 1). The network was optimized for 200 epochs with stochastic gradient descent, using a learning rate of 1×10^{-3}, a mini-batch size of 32 images and weight decay 1×10^{-6}. For the streaming method, the first four layers were streamed with tiles of 32 × 32 pixels. The network was initialized according to He *et al.*, 2015 [34].

5.3 Results on Imagenette

The loss curves of both methods (Fig. 4) were nearly identical, which empirically shows that training with streaming performed equivalently to conventional training. Small differences are likely due to losses of significance in floating point arithmetic; these differences accumulate during training and lead to small differences in loss values in later epochs.

6 Experiments on TUPAC16 Dataset

To evaluate our method on a real-world task, we used the publicly available dataset of the TUPAC16 challenge [9]. This dataset consists of 500 hematoxylin and eosin (H&E) stained whole-slide images (WSI) from breast adenocarcinoma patients. The WSIs of these patients are available from The Cancer Genome Atlas [35] together with RNA expression profiles. The expression of 11 proliferation-associated genes was combined to create one objective measure for tumor

Authorized licensed use limited to: UNIVERSITY OF BIRMINGHAM. Downloaded on April 18,2024 at 10:13:11 UTC from IEEE Xplore. Restrictions apply.

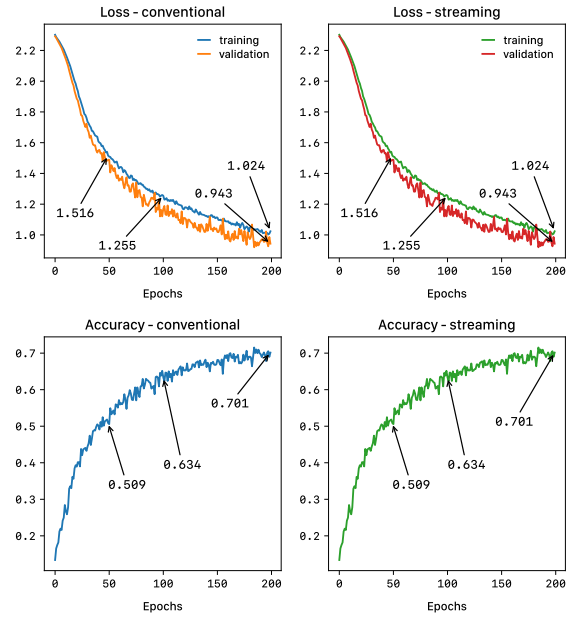

Fig. 4. Network trained from the same initialization using conventional training and streaming (dividing the input in 400 tiles of 32 × 32). Showing identical loss and accuracy curves.

growth, termed the PAM50 score [36]. This score has no known visual substrate in the images. Thus, manual labeling is considered impossible. We set aside 98 WSIs at random for tuning the algorithm and used the remaining slides for training. Additionally, an independent evaluation was performed by the challenge organizers on the test set of 321 WSIs, of which the public ground truth is not available. The submitted predictions were evaluated using Spearman's rank-order correlation between the prediction and the ground truth.

To evaluate whether CNN models can leverage and use the higher resolution information that streaming makes possible, we performed two sets of experiments to end-to-end predict the PAM50 score. For one, we trained the same model with various image sizes (1024 × 1024, 2048 × 2048, and 4096 × 4096 pixels), thus increasing input image resolution. Different networks were trained in the second set, where the depth was increased with image size (22, 25, and 28 layers for respectively 2048 × 2048, 4096 × 2096, and 8192 × 8192). By also increasing the depth, the receptive field size before the last max-pool layer is kept constant (see Table 2). All networks were trained until convergence; the checkpoint with the highest Spearman's correlation coefficient on the tuning set was submitted for independent evaluation on the test set.

6.1 Data Preparation

The images were extracted from the WSIs at image spacing $16.0\mu m$ for the 1024 × 1024 experiments, $8.0\mu m$ for 2048 × 2048, etc. (see Fig. 7). Background regions were cropped, and the resulting image was either randomly cropped or zero-padded to the predefined input size.

Since the challenge consists of a limited number of slides, we applied extensive data augmentations to increase the sample size (random rotations; random horizontal or vertical flipping; random brightness, contrast, saturation, and hue shifts; elastic transformations; and cutout [37]). For all experiments, the same hyperparameters and data preprocessing were used.

TABLE 2
Network Architecture for TUPAC16 Experiments

Layers	Kernel	Channels	Details
2x 2D convolution	3x3	32	
2D max-pool	2x2	32	
2x 2D convolution	3x3	64	
2D max-pool	2x2	64	
2x 2D convolution	3x3	128	
2D max-pool	2x2	128	
2x 2D convolution	3x3	256	
2D max-pool	2x2	256	
2x 2D convolution	3x3	512	repeated for
2D max-pool	2x2	512	field of view experiment
2x 2D convolution	3x3	512	with BatchNormalization
2D max-pool	2x2	512	
2x 2D convolution	3x3	512	with BatchNormalization
2D max-pool	input size	512	
Dropout (p=0.5)		512	
Fully connected			continuous output, without non-linearity

6.2 Network Architecture and Training Scheme

The networks (see Table 2) were trained using the Adam optimizer [38] with a learning rate of 1×10^{-4}, with the default β parameters of $\beta_1 = 0.9$, $\beta_2 = 0.999$. We applied exponential decay to the learning rate of 0.99 per epoch. As an objective, we used the Huber loss with $\Delta = 1$, also called the smooth L1 loss [39]. The mini-batch size was 16 images. A dropout layer with $p = 0.5$ was inserted before the final classification layer. The networks were initialized following He *et al.*, 2015 [34]. The images were normalized using the mean and standard deviation values of the whole training set.

Streaming was applied until the final seven layers. Since BatchNormalization breaks the local properties of chained convolutional and pooling layers, it was only used in the last part of the network. Analysis of Santurkar *et al.* [40] suggests that adding only a few BatchNormalization layers towards the end of the network smooths the loss function significantly and helps optimization.

6.3 Results on TUPAC16

The task was evaluated using Spearman's correlation coefficient between the prediction and the ground truth PAM50 proliferation scores. In both experiments, an improvement of the metric was seen with increasing input sizes.

The result of the network with the input image resolution of 4096 × 4096 approached state-of-the-art for image-level regression with a score of 0.570. Note that the first entry of the leaderboard used an additional set of manual annotations of mitotic figures and is therefore not directly comparable to our experiments.

7 Experiments on CAMELYON17 Dataset

CAMELYON17 was used to evaluate the streaming method on a classification task [32]. CAMELYON17 is a large public dataset and challenge to detect metastases of adenocarcinoma in breast tissue. The dataset consists of 500 labelled WSIs and 500 unlabeled WSIs, which were respectively used as the training and test sets. In the training set, for 450 slides image-level labels were provided, while for the remaining 50 slides dense annotations (precise delineation of the metastases)

Authorized licensed use limited to: UNIVERSITY OF BIRMINGHAM. Downloaded on April 18,2024 at 10:13:11 UTC from IEEE Xplore. Restrictions apply.

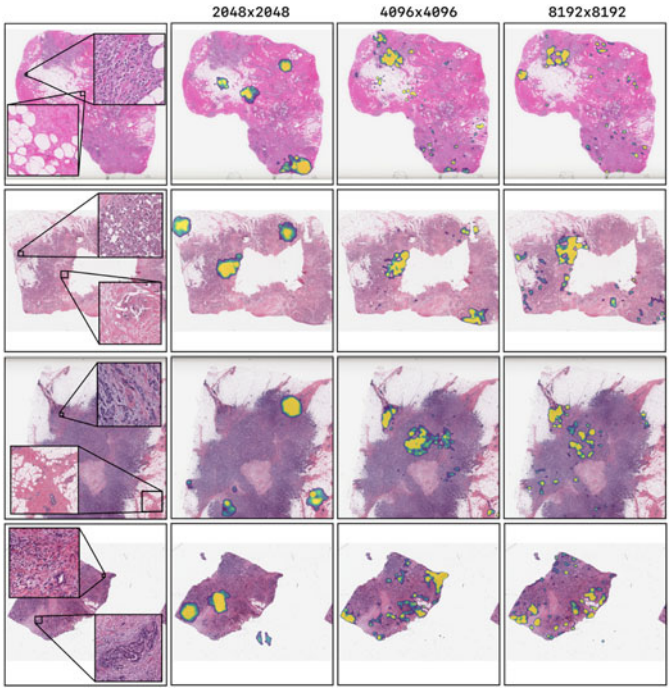

Fig. 5. Saliency maps for test set images of the TUPAC16 experiment using the best performing models. The TUPAC16 network shows highlights in cell-dense and cancerous regions. There is a trend in which the higher the input solution of the model, the less it focuses on healthy tissue. Also, higher resolution models focus on more locations of the tissue.

were supplied. The slides were collected from five different hospitals. The challenge differentiates three clinical relevant metastases types: macro-metastases ($>$ 2 mm), micro-metastases ($\leq$ 2.0 mm or $>$ 200 cells in a single cross-section), and isolated tumor cells ($\leq$ 0.2 mm or $<$ 200 cells in a single cross-section). We evaluate the slide level classification performance with multiple ROC analyses (metastasis-type versus the negative class, and negative versus all positive classes).

Data preparation for this experiment was equal to the TUPAC16 challenge. We picked 90 WSIs of the challenge training set at random to be used as our tuning set.

Confidence intervals were obtained through bootstrapping of the test set, ensuring the same sampling across the different resolutions. Furthermore, we performed a permutation test to assess statistical significance.

7.1 Network Architecture and Training Scheme

We used the same training schedule and underlying architecture as the TUPAC16 experiments. We altered the architecture by disabling dropout, and to reduce problems with exploding gradients in the beginning of the network, we replaced BatchNormalization with weight decay of 1×10^{-6} and layer-sequential unit-variance (LSUV) initialization [42]. We applied the LSUV scaling per kernel channel [43]. The mean and standard deviation per layer activation were calculated over ten mini-batches by keeping track of the sum and squares of the channels per tile during streaming; the reformulation of variance as $\mathbb{E}[X^2] - \mu^2$ was used to calculate the full standard deviation of ten mini-batches before applying LSUV.

7.2 Results on CAMELYON17

The network trained with 8192 $\times$ 8192 images is significantly better than the models trained with 4096 $\times$ 4096 images in the discriminating macro-metastases from negative cases, and significantly better than the 2048 $\times$ 2048 model in discriminating negative cases from cases with any metastasis (see Table 5).

TABLE 3
TUPAC16: Performance of the Models on the Independent Test Test, Spearman's Rho Correlation Coefficient

Experiment	Input size	Test set performance
Equal number of parameters	1024x1024	0.485 (0.441-0.527)
	2048x2048	0.491 (0.448-0.533)
	4096x4096	0.536 (0.495-0.575)
Equal field of view before global max-pool (increasing depth)	2048x2048	0.491 (0.448-0.533)
	4096x4096	0.570 (0.531-0.606)
	8192x8192	0.560 (0.520-0.597)

7.3 Saliency Maps

Saliency maps were created for the networks trained with the largest resolution (8192 $\times$ 8192 pixels) according to Simonyan *et al*. [44]. For better visualization on lower resolution, a Gaussian blur was applied with $\sigma = 50$ for 8192 $\times$ 8192 network, $\sigma = 25$ for the 4096 $\times$ 4096 models, and $\sigma = 12,5$ for the 2048 $\times$ 2048 models. Since a few gradient values can be significantly higher than others, we capped the upper gradient values at the 99th percentile [45]. The upper 50th percentile was overlayed on top of the original image (See Fig. 5).

8 Discussion and Conclusion

We presented a novel streaming method to train CNNs with tiled inputs, allowing inputs of arbitrary size. We showed that the reconstructed gradients of the neural network weights using tiles were equivalent to those obtained with non-tiled inputs.

In the first experiment on ImageNette, we empirically showed that the training behavior of our proposed streaming method was similar to the behavior in the non-streaming case. Small differences occur later in training due to loss of significance in floating-point arithmetic. These differences accumulated during training and lead to the small difference in loss values in later epochs. However, they do not seem to harm performance. Most modern frameworks have similar problems due to their use of non-deterministic operations.

The second and third experiments showed that our streaming method can train CNNs with multi-megapixel images that, due to memory requirements in the non-streaming case, would not be able to fit on current hardware. When trained using the conventional method, without streaming, the experiment with the highest-resolution images (8192×8192 pixels) would require $\sim$50 gigabytes per image, summing up to $\sim$825 gigabytes of memory per mini-batch.

Results on the TUPAC16 dataset (Table 3) showed an increasing correlation between the prediction and the proliferation score with increasing input sizes. Our 4096 $\times$ 4096 pixel network performed best. A jump in performance from 0.491 to 0.570 was seen from 2048 $\times$ 2048 to 4096 $\times$ 4096 pixels, respectively. We hypothesize that this is because tumor tissue can be discriminated from other types of tissue at these higher resolutions. However, an 8192 $\times$ 8192 pixel input size did not further improve the performance on the test set,

Authorized licensed use limited to: UNIVERSITY OF BIRMINGHAM. Downloaded on April 18,2024 at 10:13:11 UTC from IEEE Xplore. Restrictions apply.

TABLE 4
TUPAC16: Leaderboard

Experiment	Corr. coefficient
Lunit Inc., South Korea [9], [41]	0.617*
Ours (4096x4096)	**0.570**
Ours (8192x8192)	**0.560**
Tellez *et al.*, 2019 [23]	0.557
Radboud UMC Nijmegen, The Netherlands [9]	0.516
Contextvision, Sweden [9]	0.503
Belarus National Academy of Sciences [9]	0.494
The Harker School, United States [9]	0.474

**method uses additional detailed annotations from another task in the challenge and does not train a single model to predict from slide to PAM50 score.*

although the difference is minor, and the confidence interval is quite wide and overlapping. The nuclear details of cells at this resolution remain vague, which suggests that most of the information is still obtained from the morphology like in 4096 × 4096 images. Higher resolutions may be necessary to further improve performance, although we may also have hit the ceiling for the performance of this network architecture, training setup, and data. Another explanation for the lack of improvement is the increasing difficulty for the network to find the sparse information in just 400 slides using a single label or a misrepresented tuning set due to the small provided training set. As such, it is likely that for some tasks and datasets, higher-resolutions are not beneficial. Our best result on TUPAC16 approached that of the challenge winner, who used task-specific information (a network trained on mitosis detection) instead of a pure regression of one label per WSI. Our method outperformed all other methods in the challenge (see Table 4).

Results on the CAMELYON17 dataset show improvement with increasing resolution. An exception occurs for the isolated tumor cells class; even at the highest resolution applied, the CNN was unable to differentiate isolated tumor cells. To accurately identify lesions of that size, the resolution would probably need to be increased by at least a factor of four. Furthermore, this class is also underrepresented (n=31) in the provided training set. The 8192 × 8192 network was significantly better than 4096 × 4096 and 2048 × 2048 in the discriminating macro-metastases from negative cases and significantly better than 2048 × 2048 in discriminating negative cases from cases with any metastasis.

Using saliency maps, we visualized what the models would change on the input to make it more closely resemble the assigned class. These maps show us which parts of the image the model takes into consideration [44]. Saliency maps

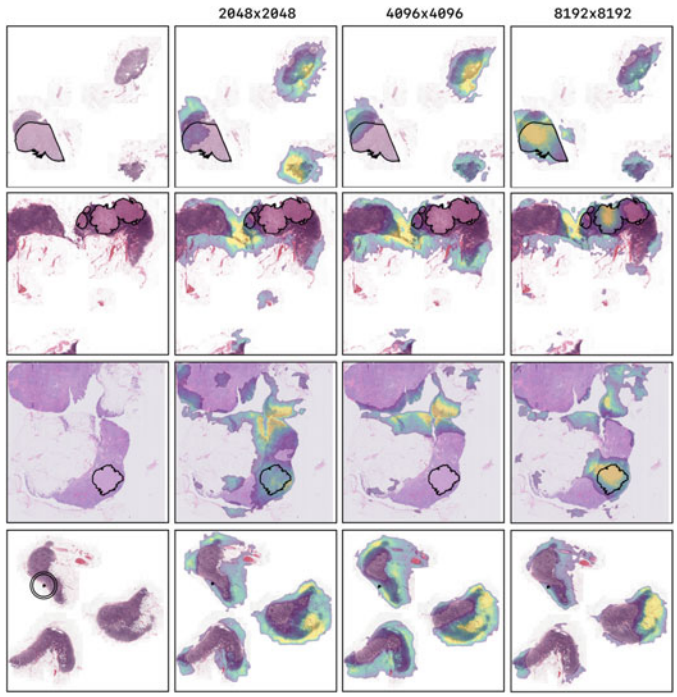

Fig. 6. Saliency maps for images of the tuning set of the CAMELYON17 experiment. The highest resolution model, trained on image-level labels shows highlights corresponding to the ground truth pixel-level annotation of a breast cancer metastasis. The lower resolution models have lower probability for the ground truth class and show little correspondence to the location of the metastases. The last row shows a micro metastasis for which models failed to recognize.

of our CNNs trained on higher resolutions suggest that the networks learn the relevant features of the high-resolution images (see Fig. 6). The image-level trained CAMELYON17 network shows highlights corresponding to the ground truth pixel-level annotation of a breast cancer metastasis. The TUPAC16 network shows highlights in cell-dense regions.

The streaming method has advantages over prior work on this topic. For streaming, we do not need to alter the dataset by resizing or creating additional pixel-level labels (which is sometimes not possible). Also, we do not need to change the usage of the dataset like in the MIL paradigm or use compression techniques. Finally, we are not limited to specific architectural choices for our network, such as in RevNet; streaming can be applied to any state-of-the-art network, such as Inception or DenseNet.

While increasing input sizes and resolutions are beneficial in various tasks, there are some drawbacks. A limitation is the increase in computation time with increasing input

TABLE 5
CAMELYON17 Results on the Independent Test Set (AUC)

Input	Negative	Isolated tumor cells	Micro-metastases	Macro-metastases
	n=260	n=35	n=83	n=122
2048^2	0.580 (0.529-0.628)	0.450 (0.363-0.539)	0.689 (0.620-0.755)	0.515 (0.451-0.577)
4096^2	0.648 (0.601-0.696, p_1=0.03)	**0.533** (0.422-0.642, p_1=0.13)	0.669 (0.602-0.733, p_1=0.65)	0.663 (0.603-0.719, p_1<0.001)
8192^2	**0.706** (0.660-0.751, p_1<0.001, p_2=0.06)	0.463 (0.359-0.569, p_1=0.43, p_2=0.83)	**0.709** (0.645-0.769, p_1=0.35, p_2=0.22)	**0.827** (0.777-0.874, p_1<0.001, p_2<0.001)

One-sided permutation test were performed between the each higher resolution and the lower resolutions (e.g., 8192^2 versus 4096^2 and 2048^2) to assess statistically significant differences. p_1 denodes a comparison against 2048^2 and p_2 against 4096^2. The 8192^2 network is significantly better than 4096^2 and 2048^2 in the discriminating macro-metastases from negative cases, and significantly better than 2048×2048 in discriminating negative cases from cases with any metastasis.

Authorized licensed use limited to: UNIVERSITY OF BIRMINGHAM. Downloaded on April 18,2024 at 10:13:11 UTC from IEEE Xplore. Restrictions apply.

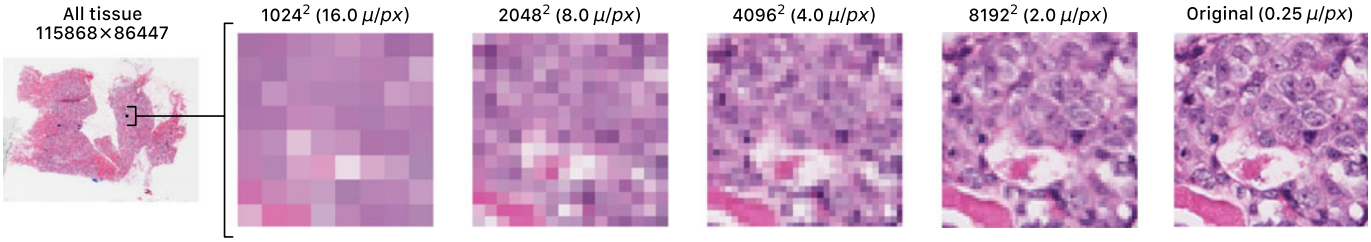

Fig. 7. Resolution examples of resized whole slide images. Example slide of TUPAC16 showing tumor, illustrating the increasing detail with increasing input size.

sizes (Fig. 3). This can be partially counteracted by dividing the batch over multiple GPUs. Due to this limitation, we did not increase resolution further in our experiments. Future research could attempt to speed up computation on the tiles, e.g., by training with mixed precision [29] or depth-wise separable convolutions [25]. One could also try to start with a pre-trained network (e.g., on ImageNet) and fine-tune for a shorter period.

Another limitation is the inability to use feature map-wide operations in the streaming part of the network, e.g., BatchNormalization. In this work, we replaced some benefits of BatchNormalization, namely the robustness against bad initialization and the regularization, with LSUV initialization and weight decay. Future work could focus on normalization techniques that retain the local properties of the relation between the output and input of the streaming part of the network, e.g., weight normalization [46].

Although this approach can, in theory, be used for segmentation, streaming a segmentation network, such as U-Net [47], will require some engineering. We would have to stream the encoder, "checkpointing" and reconstructing feature maps at the final convolution of every level, for the skip connections. Then, we would have to stream the decoding separately and carefully supply the spatially correct regions of the checkpointed skip connections to each tile. Equally for the backpropagation, reconstructing the gradients at the beginning of each decode level to make the skip connections work. This would be the case for a regular U-Net, if we would add more levels to U-Net, you will have to train middle layers without streaming, as the field of view of these layers could be too high (requiring a big overlap, making streaming less memory efficient).

Improving the performance of the high-resolution-trained networks could be a research topic of interest. In the TUPAC16 and CAMELYON17 experiments, we increased depth as we increased the input size. However, a recent work [26] – though on a maximum 480×480 image size – suggests a "compound" scaling rule in which the input resolution is scaled together with depth and width of the network.

This paper focused on streaming two-dimensional images, but since convolutions over higher-dimensional data have the same local properties, one could leverage the same technique for, for example, 3D volumetric radiological images.

Acknowledgments

The authors would like to thank Erdi Çallı for his help in proofreading the equations. This work was supported by the Dutch Cancer Society (KWF), grant number KUN 2015-797.

References

[1] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, Dec. 2015.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[4] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf.*, 2016, pp. 87.1–87.12. [Online]. Available: https://doi.org/10.5244/C.30.87
[5] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, 2009.
[6] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," 2016, *arXiv : 1604.06174*.
[7] J. Howard, "Imagenette: A smaller subset of 10 easily classified classes from ImageNet, and a little more French." [Online]. Available: https://github.com/fastai/imagenette
[8] G. Litjens *et al.*, "1399 H&E-stained sentinel lymph node sections of breast cancer patients: The CAMELYON dataset," *Gigasci.*, vol. 7, no. 6, 2018, Art. no. giy065.
[9] M. Veta *et al.*, "Predicting breast tumor proliferation from whole-slide images: The TUPAC16 challenge," *Med. Image Anal.*, vol. 54, pp. 111–121, May 2019.
[10] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, "Deep learning in remote sensing applications: A meta-analysis and review," *ISPRS J. Photogrammetry Remote Sens.*, vol. 152, pp. 166–177, Jun. 2019.
[11] G. Litjens *et al.*, "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, pp. 60–88, 2017.
[12] G. Campanella *et al.*, "Clinical-grade computational pathology using weakly supervised deep learning on whole slide images," *Nat. Med.*, vol. 25, pp. 1301–1309, Aug. 2019.
[13] P. Courtiol, E. W. Tramel, M. Sanselme, and G. Wainrib, "Classification and disease localization in histopathology using only global labels: A weakly-supervised approach," 2018, *arXiv :1802.02212*.
[14] G. Quellec, G. Cazuguel, B. Cochener, and M. Lamard, "Multiple-instance learning for medical image and video analysis," *IEEE Rev. Biomed. Eng.*, vol. 10, pp. 213–234, 2017.
[15] M. Ilse, J. M. Tomczak, and M. Welling, "Attention-based deep multiple instance learning," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 3376–3391.
[16] H. D. Couture, J. S. Marron, C. M. Perou, M. A. Troester, and M. Niethammer, "Multiple instance learning for heterogeneous images: Training a CNN for histopathology," in *Proc. Int. Conf. Med. Image Comput. Comput. Assisted Intervention*, 2018, pp. 254–262.
[17] J. D. Ianni *et al.*, "Tailored for real-world: A whole slide image classification system validated on uncurated multi-site data emulating the prospective pathology workload," *Sci. Rep.*, vol. 10, no. 1, 2020, Art. no. 3217.
[18] L. Hou, D. Samaras, T. M. Kurc, Y. Gao, J. E. Davis, and J. H. Saltz, "Patch-based convolutional neural network for whole slide tissue image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2424–2433.
[19] N. Dong, M. Kampffmeyer, X. Liang, Z. Wang, W. Dai, and E. Xing, "Reinforced auto-zoom net: Towards accurate and fast breast cancer segmentation in whole-slide images," *Deep Learn. Med. Image Anal. Multimodal Learn. Clin. Decis. Support. Lecture Notes Comput. Sci.*, vol. 11045, pp. 317–325, 2018. [Online]. Available: https://doi.org/10.1007/978-3-030-00889-5_36

Authorized licensed use limited to: UNIVERSITY OF BIRMINGHAM. Downloaded on April 18,2024 at 10:13:11 UTC from IEEE Xplore. Restrictions apply.

[20] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2204–2212.
[21] A. Katharopoulos and F. Fleuret, "Processing megapixel images with deep attention-sampling models," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 3282–3291.
[22] A. Recasens, P. Kellnhofer, S. Stent, W. Matusik, and A. Torralba, "Learning to zoom: A saliency-based sampling layer for neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 52–67.
[23] D. Tellez, G. Litjens, J. van der Laak, and F. Ciompi, "Neural image compression for gigapixel histopathology image analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Aug. 22, 2019, doi: 10.1109/TPAMI.2019.2936841.
[24] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, "The reversible residual network: Backpropagation without storing activations," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 2211–2221.
[25] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1800–1807.
[26] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.
[27] G. Huang, Z. Liu, L. Van Der Maaten , and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. 30th IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2261–2269.
[28] S. R. Bulo, L. Porzi, and P. Kontschieder, "In-place activated BatchNorm for memory-optimized training of DNNs," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5639–5647.
[29] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Proc. Deep Learn. Unsupervised Feature Learn. Workshop*, 2011.
[30] J. Zhang, S. H. Yeung, Y. Shu, B. He, and W. Wang, "Efficient memory management for GPU-based deep learning systems," 2019, *arXiv:1903.06631*.
[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[32] P. Bándi *et al.*, "From detection of individual metastases to classification of lymph node status at the patient level: The CAMELYON17 challenge," *IEEE Trans. Med. Imag.*, vol. 38, no. 2, pp. 550–560, Feb. 2019.
[33] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.
[34] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.
[35] J. N. Weinstein *et al.*, "The cancer Genome atlas pan-cancer analysis project," *Nat. Genetics*, vol. 45, pp. 1113–1120, Oct. 2013.
[36] T. O. Nielsen *et al.*, "A comparison of PAM50 intrinsic subtyping with immunohistochemistry and clinical prognostic factors in tamoxifen-treated estrogen receptor–positive breast cancer," *Clin. Cancer Res.*, vol. 16, pp. 5222–5232, Nov. 2010.
[37] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*.
[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.
[39] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.
[40] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 2483–2493.
[41] K. Paeng, S. Hwang, S. Park, and M. Kim, "A unified framework for tumor proliferation score prediction in breast histopathology," *Deep Learn. Med. Image Anal. Multimodal Learn. Clin. Decis. Support*, vol. 10553, pp. 231–239, 2017.
[42] D. Mishkin and J. Matas, "All you need is a good init," in *Proc. Int. Conf. Learn. Representations*, Nov. 2016.
[43] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell, "Data-dependent initializations of convolutional neural networks," in *Proc. Int. Conf. Learn. Representations*, Nov. 2016.
[44] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2013, *arXiv:1312.6034*.
[45] D. Smilkov, N. Thorat, B. Kim, F. B. Viégas, and M. Wattenberg, "SmoothGrad: Removing noise by adding noise," in *Proc. Int. Conf. Mach. Learn.*, 2017.
[46] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 901–909.
[47] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, 2015, pp. 234–241.

Hans Pinckaers received the MD degree in medicine from Leiden University Medical Center, Leiden, The Netherlands, in 2016. He is currently working toward the PhD degree at the Computational Pathology Group, Department of Pathology, Radboud University Medical Center, Nijmegen, the Netherlands. After his graduation, he worked one year as a Pathology resident. In 2017, he joined the Diagnostic Image Analysis Group where he works under the supervision of Geert Litjens and Jeroen van der Laak on deep learning for improved prognostics in prostate cancer.

Bram van Ginneken received the degrees in physics from the Eindhoven University of Technology, Eindhoven, the Netherlands, and Utrecht University, Utrecht, the Netherlands, and the PhD degree from the Image Sciences Institute on Computer-Aided Diagnosis in Chest Radiography, Utrecht University, Utrecht, the Netherlands, in 2001. He is currently a professor of medical image analysis at the Radboud University Medical Center and chairs the Diagnostic Image Analysis Group. He also works for Fraunhofer MEVIS in Bremen, Germany, and is a founder of Thirona, a company that develops software and provides services for medical image analysis. He has authored or coauthored more than 200 publications in international journals. He is a member of the Fleischner Society and the editorial board of the *Medical Image Analysis*. He pioneered the concept of challenges in medical image analysis.

Geert Litjens received the PhD degree in computerized detection of prostate cancer in multiparametric MRI from the Radboud University Medical Center, Nijmegen, The Netherlands. He is currently an assistant professor in computation pathology at Radboud University Medical Center. His research interest includes applications of machine learning to improve oncology diagnostics.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: UNIVERSITY OF BIRMINGHAM. Downloaded on April 18,2024 at 10:13:11 UTC from IEEE Xplore. Restrictions apply.