Improving EfficientNet for JPEG Steganalysis

Yassine Yousfi
Binghamton University
Department of Electrical and Computer Engineering
Binghamton, NY 13902-6000
yyousfi1@binghamton.edu

Jan Butora
Binghamton University
Department of Electrical and Computer Engineering
Binghamton, NY 13902-6000
jbutora1@binghamton.edu

Jessica Fridrich
Binghamton University
Department of Electrical and Computer Engineering
Binghamton, NY 13902-6000
jfridrich@binghamton.edu

Clément Fuji Tsang
NVIDIA
cfujitsang@nvidia.com

ABSTRACT

In this paper, we study the EfficientNet family pre-trained on ImageNet when used for steganalysis using transfer learning. We show that certain "surgical modifications" aimed at maintaining the input resolution in EfficientNet architectures significantly boost their performance in JPEG steganalysis, establishing thus new benchmarks. The modified models are evaluated by their detection accuracy, the number of parameters, the memory consumption, and the total floating point operations (FLOPs) on the ALASKA II dataset. We also show that, surprisingly, EfficientNets in their "vanilla form" do not perform as well as the SRNet in BOSSbase+BOWS2. This is because, unlike ALASKA II images, BOSSbase+BOWS2 contains aggressively subsampled images with more complex content. The surgical modifications in EfficientNet remedy this underperformance as well.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Image processing**.

KEYWORDS

Steganography, Steganalysis, EfficientNet, Convolutional Neural Networks, ALASKA

ACM Reference Format:
Yassine Yousfi, Jan Butora, Jessica Fridrich, and Clément Fuji Tsang. 2021. Improving EfficientNet for JPEG Steganalysis. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security (IHMMSec '21), June 22–25, 2021, Virtual Event, Belgium.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3437880.3460397

IH&MMSec '21, June 22–25, 2021, Virtual Event, Belgium

ACM ISBN 978-1-4503-8295-3/21/06…$15.00
https://doi.org/10.1145/3437880.3460397

1 INTRODUCTION

Steganalysis with machine learning has undergone an explosive development during the past five years. The previous state of the art in the form of rich (high-dimensional) media models [1–8] has quickly been replaced with convolutional neural networks (CNNs) [9–18]. The early network architectures aimed at detection of steganography have been designed by domain-area experts, which explains the inheritance of certain elements in early network designs, such as high-pass preprocessing of images [9, 10], non-random initialization by filters used in rich models [11] or discrete cosine transform (DCT) kernels [18], and use of, what appears from today's point of view as non-traditional activations, such as the thresholded linear unit [11], absolute value [10], and Gaussian activation [9]. This was driven by a firm belief of domain experts that the task of steganalysis is somehow fundamentally different from the main objective of computer vision, which is object classification. Fundamentally, though, detection of modern content-adaptive steganography is equivalent to detecting noise-like signals shaped by the content itself. It is thus not surprising that CNNs trained on computer vision tasks are a good starting point for transfer learning in steganalysis, as well as the closely related field of digital forensics [19–22].

This is confirmed by the proliferation of pre-trained models from computer vision in the recent Kaggle competitions in Camera Model Identification,[1] Deep Fake Detection,[2] and especially, in the ALASKA II [23] JPEG steganalysis challenge.[3] Many participants in ALASKA II [24–26] used the popular EfficientNet [27] pre-trained on ImageNet [28] and refined for steganalysis in the JPEG domain. Such architectures achieved markedly better performance [23, 24] than the popular SRNet [29] considered as one of the state-of-the-art CNNs for steganalysis.

In this paper, we investigate several "surgical modifications" of the EfficientNet family to further improve their performance for steganalysis while keeping in mind the computational complexity both in terms of FLOPs, the memory consumption, and the number of parameters. The main idea for the surgical modifications follows what has already been hinted at in [10, 11] and further exploited in [29], namely that decreasing the resolution of the networks in early layers via pooling or striding negatively affects their detection

[1] https://www.kaggle.com/c/sp-society-camera-model-identification
[2] https://www.kaggle.com/c/deepfake-detection-challenge
[3] https://www.kaggle.com/c/alaska2-image-steganalysis

accuracy as such operations enhance image content while suppressing the noise-like stego signal. Using the ALASKA II dataset as a benchmark, we investigate several types of surgical modifications in terms of their performance and computational complexity.

Note that we do not investigate after-the-fact model compression methods such as pruning [30, 31] or distillation [32], as these are conventionally applied after an initial training, and can be applied to any architecture.

We also study the EfficientNet family in other, aggressively downsampled image datasets, such as the BOSSbase, where the EfficientNet family does not seem to perform as well with respect to the SRNet. We attribute this drop to the aggressive subsampling of images and show that the proposed surgical modifications significantly improve EfficientNet detection accuracy.

In the next section, we introduce the notation used in this paper. Section 3 lays out the experimental setting. Section 4 describes the ImageNet pre-trained CNNs and their building blocks. Section 5 describes the proposed "surgical modifications." Section 6 studies the ImageNet pre-trained EfficientNet in other datasets. The paper is concluded in Section 7.

2 NOTATION

For consistency with the results from the ALASKA II competition, we evaluate the detectors' performance using the weighted area under the receiver operating characteristic (ROC) curve (wAUC)

$$\text{wAUC} = \int_0^1 w(P_{\text{D}}(P_{\text{FA}}))P_{\text{D}}(P_{\text{FA}})\text{d}P_{\text{FA}},$$

where $P_{\text{D}}(P_{\text{FA}})$ is the probability of detection of a stego image as a function of the probability of false alarm, which defines the ROC curve. The weighting function $w(P_{\text{D}}) \propto 2$ if $P_{\text{D}} < 0.4$ and $w(P_{\text{D}}) \propto 1$ if $P_{\text{D}} \geq 0.4$ normalizes the wAUC to be in the interval $[0, 1]$. For reference, we also occasionally report the minimum average error rate under equal priors $P_{\text{E}} = \min(P_{\text{D}}(P_{\text{FA}}) + P_{\text{FA}})$.

FLOPs is the total number of floating point operations performed to do a single forward pass using a single image input, computed using the 'fvcore' package from Facebook Research.[4] Note that only multiplications are counted while additions as well as the bias are ignored. For example, a $k \times k \times C_{\text{out}}$ convolution layer with no stride and same padding operating on a $C_{\text{in}} \times H \times W$ has FLOPs $= k^2 HWC_{\text{in}}C_{\text{out}}$.

We measure the GPU memory needed to train a model using the peak memory consumption from the 'nvidia-smi' output. To this end, we choose a batch-size of 8, a single GPU, and the other hyperparameters as detailed in 4.2. Note that the memory consumption is only an estimate and strongly depends on the implementation used. Also, it should not be confused with the total GPU memory needed since a larger batch-size is used (using data parallelism over multiple GPUs) as detailed in 4.2.

3 EXPERIMENTAL SETTING

The ALASKA II [23] dataset contains 3×25,000 different cover images compressed with quality factors 75, 90, and 95, and the same number of stego images embedded with J-UNIWARD [33], J-MiPOD [34], and UERD [35], making the training set size 4×75,000 images. The payload embedded in each image was scaled so that all images were approximately equally difficult to detect – comparatively smaller payloads were embedded in smooth images with larger payloads in highly textured or noisy images. The average payload embedded across the database was 0.4 bits per non-zero AC DCT coefficient (bpnzac). The dataset was randomly divided into three sets with 4×3×22,000, 4×3×1,000, and 4×3×2,000 images, for training, validation, and testing respectively. The splits were made compatible with those used in [24]. Note that, unlike [24], this paper does not report results using test-time augmentation (TTA).

Section 6.1 describes additional datasets used to investigate the ImageNet pre-trained models in different settings. We describe those settings within Section 6 for better readability.

[4] https://github.com/facebookresearch/fvcore

4 EFFICIENTNET FOR JPEG STEGANALYSIS

4.1 EfficientNet and SE-ResNet

Our investigation is constrained to the EfficientNet family widely used by the top competitors in the Alaska II challenge, and ultimately compared with the SE-ResNet [36] used by the winner of the competition. Other work [24] reports on the MixNet [37] architecture, which we argue achieves a very similar performance as the EfficientNet due to strong architecture similarities. Thus, we do not report results using the MixNet family for this reason but expect our contributions to transfer to such similar architectures.

The EfficientNet is built using the Inverted Residual Block [38] (IR) depicted in Figure 1. The lightweight depth-wise separable convolution (D-Conv) is the key to the network's efficiency. On the other hand, the SE-ResNet uses the classical ResNet Block [39]. Both architectures use the Squeeze & Excite Block [36] with different reduction parameters.

4.2 Transfer learning procedure

Fine-tuning ImageNet pre-trained models on a steganalysis task is done using multi-class cross-entropy loss and the AdamW optimizer with 10^{-2} weight decay, for 60 epochs using a cosine learning rate scheduler with a start LR of 10^{-3} and end LR of 2×10^{-5}, and D_4 training augmentation. The model is converted and trained in Automatic Mixed Precision (AMP). We used a minimum batch size of 24, which was increased for smaller architectures to speed up training. The mini-batches were not pair-constrained, which means that on average, one batch included 1/4 cover images and 3/4 stego images randomly sampled. The JPEG images were decompressed to RGB color space without rounding or clipping. After training, we chose the best checkpoint based on the wAUC metric on the validation set.

5 "SURGICAL MODIFICATIONS" IMPROVE EFFICIENTNET

The Alaska II Kaggle competition has shown that many successful ImageNet pre-trained CNNs can achieve a very competitive performance when fine-tuned on a steganalysis task. Moreover, carefully modified ImageNet pre-trained CNNs can be made even

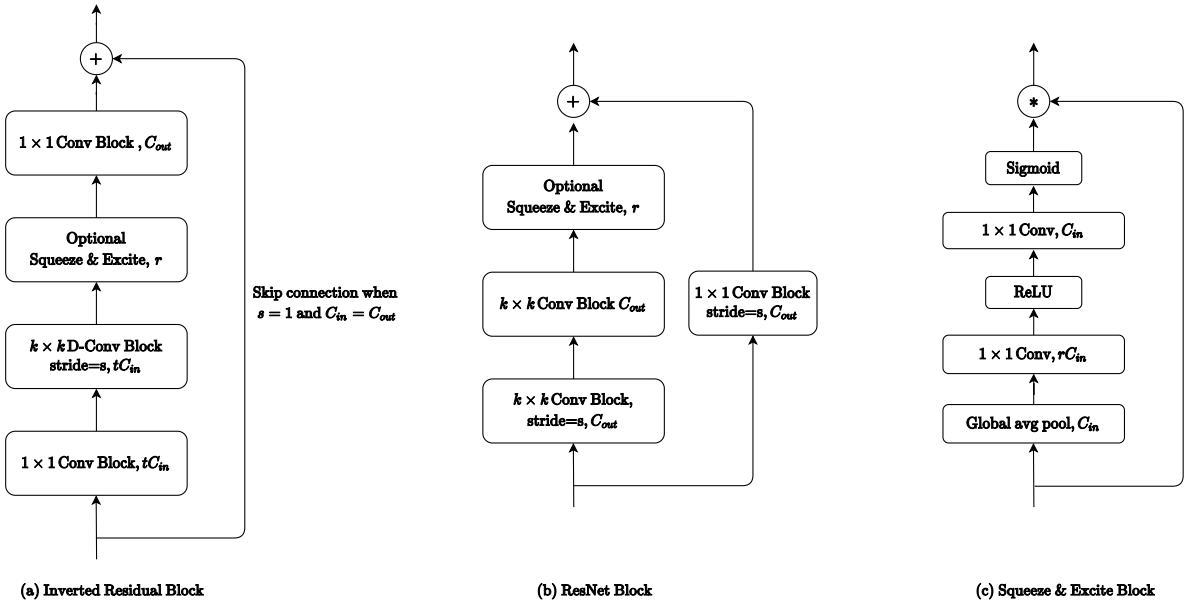

Figure 1: (a) The Inverted Residual block used in the EfficientNet architecture, t denotes the expansion parameter of the block. (b) The ResNet Block used in SE-ResNet18. (c) The Squeeze & Excite block, r denotes the reduction parameter. In (a) and (b) Conv Blocks are composed of a Convolution layer, Batch Normalization layer, and an Activation.

better. We consider two types of "surgical modifications": ablation of downsampling elements in the architectures (stride, pooling) and insertion of additional layers operating on high-resolution feature representations.

5.1 Stem stride ablation

In [24], the authors show that the performance of the MixNet-S can be significantly improved by removing the stride from its stem. This was also reported by many competitors.[5] We report similar results with the EfficientNet family for consistency with the paper's experiments. Table 1 shows that removing strides from the stem (no stride L0) and from the next strided block (no stride L2) of EfficientNet B0 significantly improves the performance but comes at a substantial price in FLOPs and memory requirements because the convolutions will operate on 4× larger volumes after each removed stride. Removing strides from the stem of EfficientNet B4 gives a stellar performance but with an unreasonably high memory consumption making the training extremely slow.

For SE-ResNet, we first remove the max-pooling in the stem and train for 10 epochs, then remove the stride and continue the training. Note that this was not described by the winner of the Alaska II competition in [26] but was essential to successfully train the modified architecture with the hyper-parameters of our experiments. Similarly, this surgical modification comes at a substantial price in FLOPs.

[5] https://www.kaggle.com/c/alaska2-image-steganalysis/discussion

Surgical modification	wAUC	FLOPs (B)	Params (M)	Mem (MiB)
Vanilla B0	.92601	2.15	4.01	3,354
B0 no stride L0	.92844	8.31	4.01	10,162
B0 no stride L2	.93552	30.73	4.01	24,184
B4 no stride L0	.94408	31.63	17.56	21,484
B6 no stride L0	.94618	69.97	40.74	40,186

Table 1: wAUC, FLOPs, and the number of parameters of different modifications of the EfficientNet family.

Surgical modification	wAUC	FLOPs (B)	Params (M)	Mem (MiB)
Vanilla SE-ResNet	.87661	9.53	11.26	3,084
SE-ResNet no stride/pool L0	.94231	144.89	11.26	8,468

Table 2: wAUC, FLOPs, and the number of parameters of the SE-ResNet-18 and its modified version.

Note that disabling stride and/or pooling in the stem does not change the number of parameters of a network, however the computational cost (GPU memory and FLOPs) to train it increase significantly. This motivates our choice to use FLOPs as a model-complexity measure.

Surgical modification	wAUC	FLOPs (B)	Params (M)	Mem (MiB)
Vanilla B0	.92601	2.15	4.01	3,354
B0 - original pre-stem	.92902	7.16	4.04	3,978
B0 - pre-stem IR blocks	.93300	4.98	4.03	6,460
B0 - post-stem IR blocks	.93313	4.88	4.02	6,812

Table 3: wAUC, FLOPs, and the number of parameters of different modifications of the EfficientNet-B0.

Surgical modification	wAUC	FLOPs (B)	Params (M)	Mem (MiB)
B0 - post-stem IR blocks $t = 1$	.93313	4.88	4.02	6,812
B0 - post-stem IR blocks $t = 4$	.93506	12.15	4.05	13,840
B0 - post-stem ResNet blocks	.93623	18.31	4.08	6,488

Table 4: wAUC, FLOPs, and the number of parameters of different variants of the post-stem surgical modification with the EfficientNet-B0.

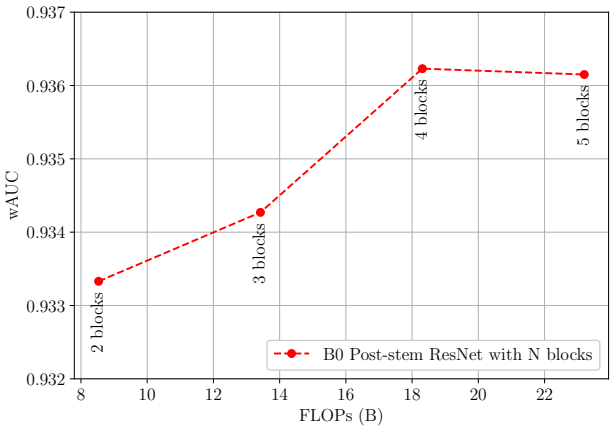

Figure 2: wAUC vs. FLOPs of EfficientNet-B0 with a post-stem modification and a varying number of inserted convolutional blocks.

5.2 Unpooled layers implant

In this section, we show that the performance can be improved at a much lower cost in term of FLOPs and memory requirements by inserting layers at influential parts of the architecture, namely in early layers to mimic the "unpooled layers" of steganalysis CNNs, such as the SRNet.

Some Alaska II competitors reported performance increase when adding layers to the EfficientNet architecture [40]. We call this surgical modification "pre-stem insertion," since the layers are implanted before the stem. The original modification described in [40] only included 3 convolutional blocks with an increased number of channels. We show that using 4 blocks and a large number of channels from the first block is beneficial. Note that we did not modify the last two layers as described in [40] as this degraded the performance in our experiments.

Additionally, we introduce a new surgical modification called "post-stem" insertion, which (i) disables the downsampling operation in the stem and (ii) inserts convolutional blocks after the stem, last of which has a stride of 2. In essence, post-stem and pre-stem are very similar architectures, with post-stem being more computationally efficient and more accurate as shown in Table 3.

We choose to study the post-stem surgery further by changing some of its design hyper-parameters. Increasing the expansion parameter t of the implanted Inverted Residual blocks improves the representation capacity of those layers by allowing to form a larger number of noise residuals. Using the ResNet blocks instead of the lightweight Inverted Residual blocks also allows the implants to form more complex residuals. Both changes improve the performance as shown in Table 4. These improvements prove, yet again, the importance of the early unpooled layers in the architectures. Note that these improvements also come at a FLOPs cost, making the surgically modified model more computationally demanding but still having a better performance-memory to compute trade-off than the stride ablation studied in Section 5.1.

We also study the effects of the number of inserted layers in Figure 2, which shows that the performance increases with increasing number of blocks, then saturates at 4 inserted ResNet blocks. He hypothesize that this optimal number of inserted layers depends on the cover and/or stego source and would require to be validated accordingly. However, for the sake of the experiments in this paper, we fix a number of inserted layers of 4 for the post-stem surgery, keeping in mind that this number might be different for different scenarios (e.g. spatial domain steganalysis).

Figure 3 describes all insertion strategies proposed and studied in this paper. The insertion strategies are shown with 4 added blocks, determined by the results shown in Figure 2. We show the performance of the modifications studied in Figure 4, removing the stride performs best but at a significant memory cost as discussed in Section 5.1, the next best modification coming at a lower memory cost is the post-stem with ResNet blocks. The SE-ResNet18 with stride and pooling removed has a significant FLOPs count due to the use of the expensive ResNet blocks, but has a reasonable memory footprint thanks to its reduced size. A similar trend is observed with P_E as a metric instead of the wAUC. The equivalent of Figure 4 with the P_E metric is shown in Figure 7 in the Appendix for reference.

5.3 Do we gain from ImageNet pre-training of modified CNNs?

A relevant question regarding the surgical modifications is whether we might gain from pre-training the modified architectures on ImageNet instead of only modifying them at the transfer learning stage. This question is especially relevant for the post-stem modification where randomly initialized layers are inserted within a network already trained on ImageNet. Table 5 shows that there is no substantial benefit from training the modified EfficientNet on ImageNet. The table also includes a "control" vanilla EfficientNet-B0 pre-training to show that our local ImageNet pre-training matches

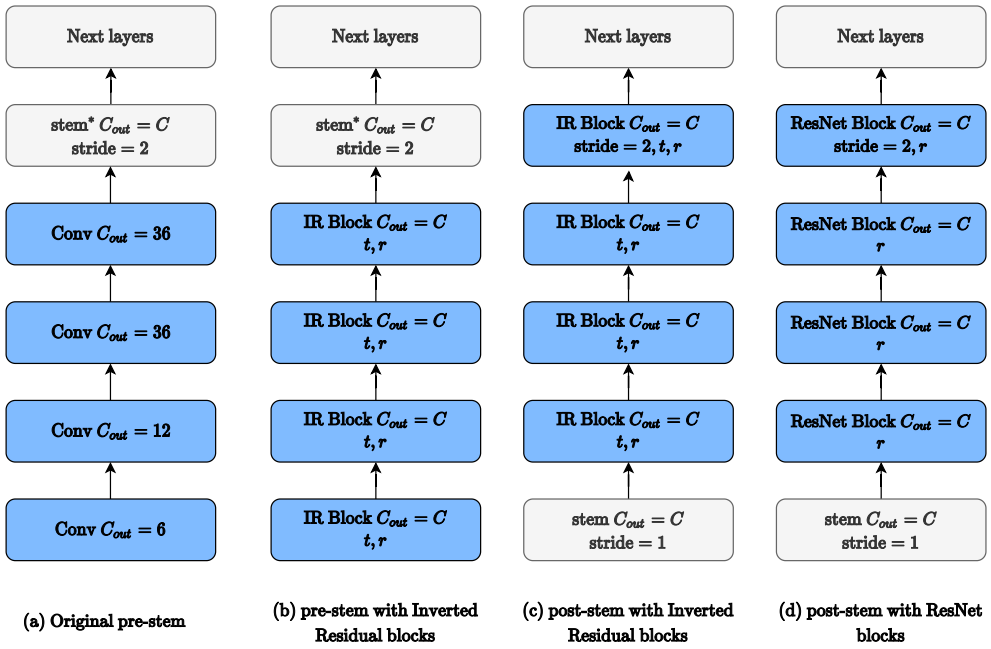

Figure 3: All surgical modifications studied. Blue blocks are the inserted blocks. In (a) and (b) the stem* convolutional kernels are duplicated and concatenated to match the previous layer's dimension.

	Pre-trained w/ modification	Surgically modified
B0 - post-stem	.93391	.93313
(control) Vanilla B0	.92609	.92601

Table 5: wAUC of EfficientNet-B0 with post-stem with Inverted Residual Block $t = 1$ and unchanged. Modification done at the pre-training stage or at the transfer learning stage.

the one done by the EfficientNet-pytorch[6] library in terms of performance in the downstream task. This means that it is safe to "surgically" apply the modifications at the transfer learning stage. Pre-training on the ImageNet dataset was done for 100 epochs with the SGD optimizer with 0.9 momentum, a weight-decay of 10^{-5}, the OneCycle learning rate scheduler with a maximum learning rate of 0.5, a minimum of 10^{-3}, and a batch-size of 512.

Figure 5 shows the validation wAUC at different training epochs of the two versions of the B0 - post-stem in Table 5. We note that while the surgically modified network starts lower than the the one pre-trained with the modification, the two versions eventually converge to a very close peak performance as shown in Table 5. The early epochs of post-stem surgically modified networks usually exhibit a lower performance due to the randomly initialized convolutional blocks inserted, but the performance increases given enough epochs.

[6] https://github.com/lukemelas/EfficientNet-PyTorch

6 WHERE DOES EFFICIENTNET SHINE?

6.1 Additional experimental setting

For this section, we use two more datasets to compare the effectiveness of the ImageNet pre-trained EfficientNet in different settings.

BOSSbase+BOWS2 is the union of BOSSbase 1.01 [41] and BOWS2 [42] converted to grayscale and resized to 256×256 using Matlab's 'imresize' with default parameters. We use JPEG quality factors 75, 90, and 95 and embedding schemes J-UNIWARD, J-MiPOD, and UERD at 0.4, 0.4 and 0.2 bnzac respectively (fixed payload sender). The dataset is randomly divided into three sets with $4 \times 3 \times 14,000$ (BOSSbase+BOWS2), $4 \times 3 \times 1,000$ (BOSSbase), $4 \times 3 \times 5,000$ images (BOSSbase) for training, validation, and testing respectively. The splits are also made compatible with [29]. Note that for consistency with the results from the ALASKA II competition, we used the same versions of the previously listed embedding schemes as in the competition's dataset. New versions of the J-MiPOD [43] or correctly implemented UERD were not considered.

We create a new dataset called ALASKA II BOSS-style which contains raw images from the ALASKA II dataset, processed using the BOSSbase processing script and resized to 256×256 using ImageMagick's resize with default parameters. We use the same embedding script as the ALASKA II dataset with an average payload across database of 0.2 bpnzac (Detectability Limited Sender [44]). The dataset is divided into the same splits as the ALASKA II dataset.

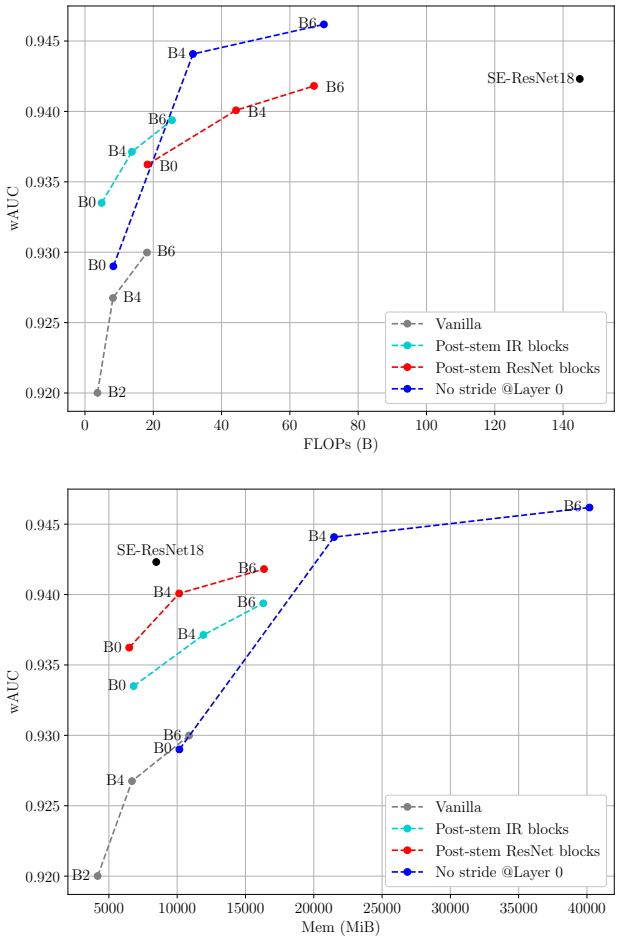

Figure 4: wAUC vs. FLOPs and memory requirements of different surgical modifications. The EfficientNet B6 with stride disabled and B6 post-stem with ResNet blocks achieve a comparable performance to the SE-ResNet18 with pool and stride disabled, with one half of the FLOPs. For reference, we include the vanilla versions of the EfficientNet trained using the schedule described in [24] in Section II.A.

6.2 Training and transfer learning procedure

We use the SRNet [29] without the pair-constraint [24] as a baseline to evaluate EfficientNet in these additional datasets. SRNet was first trained on QF75 with the pair-constraint for 200 epochs then refined on QF75, 90, and 95 without the pair-constraint for another 100 epochs. Training was done with the multi-class cross-entropy loss, using the Adamax optimizer with a 10^{-4} weight decay, the OneCycle learning rate scheduler with a start LR of 4×10^{-5}, a maximum LR of 10^{-3}, and an end LR of 2×10^{-5}. Inputs were also transformed to RGB for color images without rounding or clipping, and divided by 255 before feeding to the network.

For the ImageNet pre-trained EfficientNet and SE-ResNet18, we used the same hyper-parameters as in 4.2. For grayscale BOSSbase+BOWS2, we insert a 1×1 convolution layer with 3 output

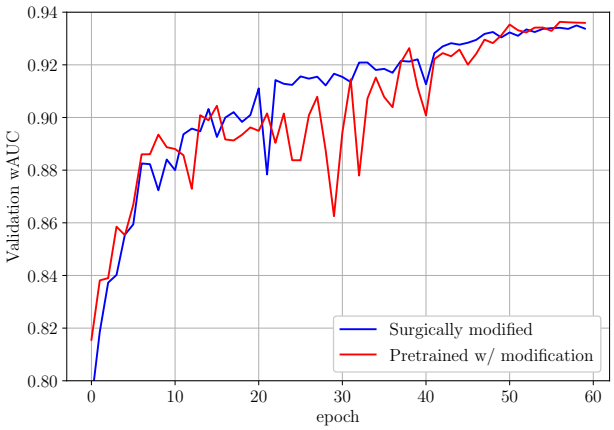

Figure 5: Validation wAUC at different training epochs of the EfficientNet-B0 with post-stem with Inverted Residual Block $t = 1$ both pre-trained with the modification or surgically modified.

channels before the stem to match the input channels of the pre-trained stem.

6.3 The effect of BOSS-style processing

This investigation section started by noticing the surprising difference between the first two columns in Table 6: in the ALASKA II dataset, the vanilla EfficientNet-B4 outperforms SRNet, but in the BOSSbase+BOWS2 dataset, SRNet outperforms the EfficientNet-B4. Note that this was also observed in [24] with the ALASKA I [45] dataset.

In [24], the authors hypothesized that this shift is due to the fact that ImageNet pre-trained models might be more data efficient than the SRNet trained from scratch.

However, we show that the main differences between the first two columns of Table 6 are mostly due to the cover processing pipeline because using a "BOSSbase-style" ALASKA II processed database again shows EfficientNet-B4 underperforming.

Table 6 shows that vanilla versions of ImageNet pre-trained models underperform in strongly subsampled cover sources, such as BOSSbase+BOWS2 and BOSS-style datasets. We hypothesize that this is due to their lack of unpooled layers. Strongly subsampled cover sources exhibit more high frequency content, which will require more layers operating at the original resolution. The proposed surgical modifications help mitigate this effect as they introduce more unpooled layers into the architecture.

Note that this observation does not hold for the UERD embedding scheme. This is because the ALASKA II competition used a faulty implementation of UERD that concentrates most of the embedding changes at the image boundary and thus is much less dependent on the cover source.

We verify that BOSS-style datasets have indeed more high frequency content by computing the average energy of the KB [46] residual over 500 images from the original ALAKA II and the ALASKA II BOSS-style. Figure 6 shows the distributions of the

Dataset	ALASKA II			BOSSbase+BOWS2			ALASKA II BOSS-style		
Sender	DeLS 0.4 J-UNI	DeLS 0.4 J-MiPOD	DeLS 0.4 UERD	PLS 0.4 J-UNI	PLS 0.4 J-MiPOD	PLS 0.2 UERD	DeLS 0.2 J-UNI	DeLS 0.2 J-MiPOD	DeLS 0.2 UERD
SRNet	.8718	.9427	.9364	.9634	.9780	.9773	.9903	.9142	.9279
B4	.8783	.9475	.9540	.9528	.9723	.9812	.9892	.9021	.9436
B4 no stride L0	**.9092**	.9592	**.9636**	.9700	.9828	.9857	.9925	.9302	.9528
B4 post-stem ResNet blocks	.9006	.9574	.9620	**.9746**	.9852	**.9879**	.9923	.9302	**.9532**
B6 post-stem ResNet blocks	.9057	.9566	.9629	.9731	**.9880**	.9842	**.9927**	**.9382**	.9503
SE-ResNet18 no stride/pool L0	.9045	**.9610**	.9611	.9729	.9839	.9865	.9911	.9234	.9491

Table 6: wAUC of different modifications of the EfficientNet, the SE-ResNet18, and the SRNet as a baseline in three different datasets with their respective sender strategies.

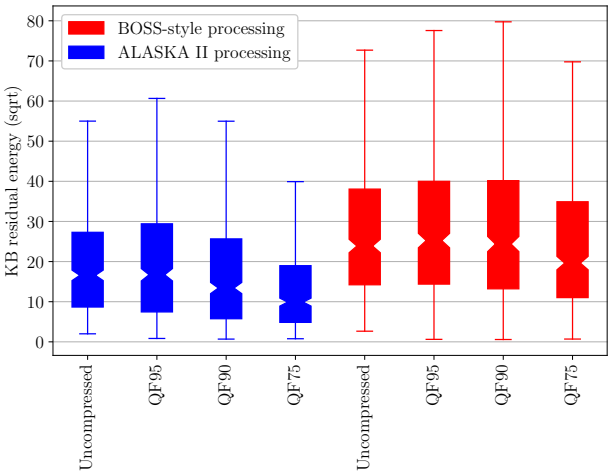

Figure 6: Distribution of the square root of the average KB residual energy across 500 images uncompressed and JPEG compressed with qualities 95, 90, and 75 from the ALASKA II and the ALASKA II BOSS-style datasets.

square root of the average energy (the square root helps avoid large outliers) for the two datasets in the uncompressed format, JPEG compressed with qualities 95, 90, and 75. Figure 6 shows that indeed, BOSS-style processing contains complex content when measured by the inability to predict the pixel values from their neighborhoods using the KB filter.

Table 6 also shows that the EfficientNet B6 (amd B4) post-stem with ResNet blocks have a very comparable performance to the SE-ResNet18 no stride/pool L0 (even slightly better) on the BOSSbase+BOWS2 and ALASKA II BOSS-style datasets. Note that on these datasets, the post-stem surgery performs better than the stride ablation, unlike in the ALASKA II dataset as shown in Figure 4.

7 CONCLUSION

We propose and study several different ways to modify the EfficientNet architecture to significantly improve performance for JPEG steganalysis. These so called "surgical modifications" are done at the transfer-learning stage to substantially improve the performance upon the original (vanilla) EfficientNet architectures. The post-stem modification boosts the performance while keeping the computational cost and the memory requirements reasonable by increasing the number of unpooled layers in the architecture. Removing the stride in the stem of the EfficientNet architectures, on the other hand, requires large GPU memory for training. The modified models reach state-of-the-art performance with less than 1/2 of the FLOPs of the current best model on the ALASKA II dataset.

We also test the EfficientNet family in different datasets and notice that in strongly subsampled cover sources (e. g., BOSSbase+BOWS2), they underperform with respect to the SRNet due to their lack of unpooled layers. The proposed surgically modified EfficientNet architectures overcome this issue and surpass the popular SRNet on a variety of datasets.

More broadly, this paper confirms that off-the-shelf successful computer vision architectures, such as the EfficientNet, can reach unparalleled performance in JPEG steganalysis. No special elements were added to the architecture besides the unpooled layers known to be beneficial for steganalysis.

ACKNOWLEDGMENTS

The work on this paper was supported by NSF grants no. 1561446 and 2028119.

REFERENCES

[1] J. Fridrich and J. Kodovský, "Rich models for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 868–882, June 2011.

[2] J. Kodovský and J. Fridrich, "Steganalysis of JPEG images using rich models," in *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2012*, A. Alattar, N. D. Memon, and E. J. Delp, Eds., vol. 8303, San Francisco, CA, January 23–26, 2012, pp. 0A 1–13.

[3] L. Chen, Y. Shi, P. Sutthiwan, and X. Niu, "A novel mapping scheme for steganalysis," in *International Workshop on Digital Forensics and Watermaking*, ser. LNCS, Y. Shi, H.-J. Kim, and F. Perez-Gonzalez, Eds., vol. 7809. Springer Berlin Heidelberg, 2013, pp. 19–33.

[4] W. Tang, H. Li, W. Luo, and J. Huang, "Adaptive steganalysis against WOW embedding algorithm," in *2nd ACM IH&MMSec. Workshop*, A. Uhl, S. Katzenbeisser, R. Kwitt, and A. Piva, Eds., Salzburg, Austria, June 11–13, 2014, pp. 91–96.

[5] T. Denemark, V. Sedighi, V. Holub, R. Cogranne, and J. Fridrich, "Selection-channel-aware rich model for steganalysis of digital images," in *IEEE International Workshop on Information Forensics and Security*, Atlanta, GA, December 3–5, 2014.

[6] V. Holub and J. Fridrich, "Phase-aware projection model for steganalysis of JPEG images," in *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2015*, A. Alattar and N. D. Memon, Eds., vol. 9409, San Francisco, CA, February 8–12, 2015.

[7] X. Song, F. Liu, C. Yang, X. Luo, and Y. Zhang, "Steganalysis of adaptive JPEG steganography using 2D Gabor filters," in *3rd ACM IH&MMSec. Workshop*, P. Comesana, J. Fridrich, and A. Alattar, Eds., Portland, Oregon, June 17–19, 2015.
[8] T. Denemark, M. Boroumand, and J. Fridrich, "Steganalysis features for content-adaptive JPEG steganography," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1736–1746, August 2016.
[9] Y. Qian, J. Dong, W. Wang, and T. Tan, "Deep learning for steganalysis via convolutional neural networks," in *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2015*, A. Alattar and N. D. Memon, Eds., vol. 9409, San Francisco, CA, February 8–12, 2015.
[10] G. Xu, H. Z. Wu, and Y. Q. Shi, "Structural design of convolutional neural networks for steganalysis," *IEEE Signal Processing Letters*, vol. 23, no. 5, pp. 708–712, May 2016.
[11] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2545–2557, November 2017.
[12] M. Yedroudj, F. Comby, and M. Chaumont, "Yedroudj-net: An efficient CNN for spatial steganalysis," in *IEEE ICASSP*, Alberta, Canada, April 15–20, 2018, pp. 2092–2096.
[13] B. Li, W. Wei, A. Ferreira, and S. Tan, "ReST-Net: Diverse activation modules and parallel subnets-based CNN for spatial image steganalysis," *IEEE Signal Processing Letters*, vol. 25, no. 5, pp. 650–654, May 2018.
[14] J. Zeng, S. Tan, G. Liu, B. Li, and J. Huang, "WISERNet: Wider separate-then-reunion network for steganalysis of color images," *CoRR*, vol. abs/1803.04805, 2018. [Online]. Available: http://arxiv.org/abs/1803.04805
[15] M. Chen, V. Sedighi, M. Boroumand, and J. Fridrich, "JPEG-phase-aware convolutional neural network for steganalysis of JPEG images," in *The 5th ACM Workshop on Information Hiding and Multimedia Security*, M. Stamm, M. Kirchner, and S. Voloshynovskiy, Eds., Philadelphia, PA, June 20–22, 2017.
[16] J. Yang, Y.-Q. Shi, E. Wong, and X. Kang, "JPEG steganalysis based on DenseNet," vol. abs/1711.09335, 2017. [Online]. Available: http://arxiv.org/abs/1711.09335
[17] J. Zeng, S. Tan, B. Li, and J. Huang, "Large-scale JPEG image steganalysis using hybrid deep-learning framework," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1200–1214, May 2018.
[18] G. Xu, "Deep convolutional neural network to detect J-UNIWARD," in *The 5th ACM Workshop on Information Hiding and Multimedia Security*, M. Stamm, M. Kirchner, and S. Voloshynovskiy, Eds., Philadelphia, PA, June 20–22, 2017.
[19] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Niessner, "Faceforensics++: Learning to detect manipulated facial images," in *2019 IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1–11.
[20] F. Marra, D. Gragnaniello, D. Cozzolino, and L. Verdoliva, "Detection of GAN-generated fake images over social networks," in *2018 IEEE Conference on Multimedia Information Processing and Retrieval*, 2018, pp. 384–389.
[21] D. Cozzolino, J. Thies, A. Rössler, C. Riess, M. Nießner, and L. Verdoliva, "ForensicTransfer: Weakly-supervised domain adaptation for forgery detection," *arXiv*, 2018. [Online]. Available: http://arxiv.org/abs/1812.02510
[22] S. Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "CNN-generated images are surprisingly easy to spot... for now," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8692–8701.
[23] R. Cogranne, Q. Giboulot, and P. Bas, "ALASKA–2: Challenging academic research on steganalysis with realistic images," in *IEEE International Workshop on Information Forensics and Security*, Held virtually, December 6–11, 2020.
[24] Y. Yousfi, J. Butora, E. Khvedchenya, and J. Fridrich, "ImageNet pre-trained CNNs for JPEG steganalysis," in *IEEE International Workshop on Information Forensics and Security*, Held virtually, December 6–11, 2020.
[25] K. Chubachi, "An ensemble model using CNNs on different domains for ALASKA2 image steganalysis," in *IEEE International Workshop on Information Forensics and Security*, Held virtually, December 6–11, 2020.
[26] G. Xu, "1st place solution," https://www.kaggle.com/c/alaska2-image-steganalysis/discussion/168548, 2020, [Online; accessed 29-December-2020].
[27] T. Mingxing and V. L. Quoc, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, June 9–15, 2019, pp. 6105–6114.
[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 20–25, 2009, pp. 248–255.
[29] M. Boroumand, M. Chen, and J. Fridrich, "Deep residual network for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1181–1193, May 2019.
[30] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, 2015.
[31] S. Tan, W. Wu, Z. Shao, Q. Li, B. Li, and J. Huang, "CALPA-NET: Channel-pruning-assisted deep residual network for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 131–146, 2021.
[32] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
[33] V. Holub, J. Fridrich, and T. Denemark, "Universal distortion design for steganography in an arbitrary domain," *EURASIP Journal on Information Security, Special Issue on Revised Selected Papers of the 1st ACM IH and MMS Workshop*, vol. 2014:1, 2014.
[34] R. Cogranne, Q. Giboulot, and P. Bas, "Steganography by minimizing statistical detectability: The cases of JPEG and color images," in *The 8th ACM Workshop on Information Hiding and Multimedia Security*, C. Riess and F. Schirrmacher, Eds. Held virtually: ACM Press, 2020.
[35] L. Guo, J. Ni, and Y. Q. Shi, "Uniform embedding for efficient JPEG steganography," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 5, pp. 814–825, May 2014.
[36] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.
[37] T. Mingxing and V. L. Quoc, "MixConv: Mixed depthwise convolutional kernels," in *British Machine Vision Conference, BMVC*, September 9–12, 2019.
[38] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.
[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 27–30, 2016, pp. 770–778.
[40] Q. Ha, "Single effnet-b0 private LB 0.921: How to modify effnet architecture," https://www.kaggle.com/c/alaska2-image-steganalysis/discussion/168542, 2020, [Online; accessed 29-December-2020].
[41] P. Bas, T. Filler, and T. Pevný, "Break our steganographic system – the ins and outs of organizing BOSS," in *Information Hiding, 13th International Conference*, ser. Lecture Notes in Computer Science, T. Filler, T. Pevný, A. Ker, and S. Craver, Eds., vol. 6958, Prague, Czech Republic, May 18–20, 2011, pp. 59–70.
[42] P. Bas and T. Furon, "BOWS-2," http://bows2.ec-lille.fr, July 2007.
[43] R. Cogranne, "J-MiPOD source code," personal communication, [e-mail; sent 21-December-2020].
[44] R. Cogranne, V. Sedighi, and J. Fridrich, "Practical strategies for contentadaptive batch steganography and pooled steganalysis," in *Proceedings IEEE, International Conference on Acoustics, Speech, and Signal Processing*, New Orleans, LA, March 5–9, 2017, pp. 2122–2126.
[45] R. Cogranne, Q. Giboulot, and P. Bas, "The ALASKA steganalysis challenge: A first step towards steganalysis "Into the wild"," in *The 7th ACM Workshop on Information Hiding and Multimedia Security*, R. Cogranne and L. Verdoliva, Eds. Paris, France: ACM Press, July 3–5, 2019.
[46] A. D. Ker and R. Böhme, "Revisiting weighted stego-image steganalysis," in *Proceedings SPIE, Electronic Imaging, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, E. J. Delp, P. W. Wong, J. Dittmann, and N. D. Memon, Eds., vol. 6819, San Jose, CA, January 27–31, 2008, pp. 5 1–17.

8 APPENDIX

We show the details of the performance of all architectures studied in this paper for every stego scheme and JPEG quality factor in the ALASKA II dataset.

Backbone	Model Surgical Modification	UERD QF75	UERD QF90	UERD QF95	J-UNIWARD QF75	J-UNIWARD QF90	J-UNIWARD QF95	J-MiPOD QF75	J-MiPOD QF90	J-MiPOD QF95	Mixture	FLOPs (B)	Params (M)	Mem (MiB)
B0	None	.9513	.9653	.9517	.8766	.8777	.8847	.9820	.9740	.8532	.92601	2.15	4.01	3,354
	No stride L0	.9529	.9640	.9539	.8753	.8830	.8881	.9811	.9793	.8630	.92844	8.31	4.01	10,162
	No stride L2	.9539	.9637	.9550	.8892	.8944	.9054	**.9856**	.9839	.8779	.93552	30.73	4.01	24,184
	Original pre-stem	.9515	.9623	.9481	.8768	.8846	.8941	.9819	.9785	.8701	.92902	7.16	4.04	3,978
	Pre-stem IR blocks $t = 1$	.9552	**.9662**	.9553	.8866	.8915	.8967	.9840	.9792	.8686	.93300	4.98	4.03	6,460
	Post-stem IR blocks $t = 1$	.9532	.9649	.9563	.8854	.8925	.9003	.9818	.9794	.8735	.93313	4.88	4.02	6,812
	Post-stem IR blocks $t = 4$	.9523	.9640	**.9569**	.8904	.8973	.9048	.9834	.9815	.8738	.93506	12.15	4.05	13,840
	Post-stem ResNet blocks	**.9552**	.9650	.9559	**.8932**	**.9004**	**.9066**	.9830	**.9818**	**.8746**	**.93623**	18.31	4.08	6,488
B4	None	.9542	.9624	.9497	.8783	.8788	.8860	.9805	.9759	.8596	.92675	8.20	17.56	6,692
	No stride L0	.9608	**.9699**	**.9631**	**.9069**	**.9116**	**.9149**	**.9879**	**.9851**	**.8853**	**.94408**	31.63	17.56	21,484
	Post-stem IR $t = 1$	.9587	.9671	.9578	.8956	.8996	.9067	.9810	.9790	.8773	.93713	13.75	17.57	11,924
	Post-stem ResNet blocks	**.9620**	.9692	.9585	.8971	.9037	.9097	.9858	.9822	.8821	.94008	44.23	17.72	10,146
B6	None	.9534	.9644	.9514	.8848	.8901	.8920	.9808	.9764	.8601	.92998	18.16	40.74	10,868
	No stride L0	**.9625**	**.9715**	**.9628**	**.9093**	**.9139**	**.9168**	**.9912**	**.9871**	**.8881**	**.94618**	69.97	40.74	40,186
	Post-stem IR $t = 1$	.9590	.9685	.9542	.8986	.9036	.9072	.9850	.9830	.8841	.93938	25.48	40.77	16,304
	Post-stem ResNet blocks	**.9625**	.9707	.9597	.9054	.9073	.9138	.9836	.9813	.8817	.94181	67.07	40.98	16,356
SE-ResNet18	None	.9321	.9363	.9298	.8019	.7668	.7534	.9687	.9549	.7615	.87661	9.53	11.26	3,084
	No stride/pool L0	**.9621**	**.9686**	**.9570**	**.8999**	**.9104**	**.9116**	**.9864**	**.9853**	**.8881**	**.94231**	144.89	11.26	8,468

Table 7: wAUC, FLOPs, memory, and the number of parameters of different architectures and surgical modifications in the ALASKA II dataset.

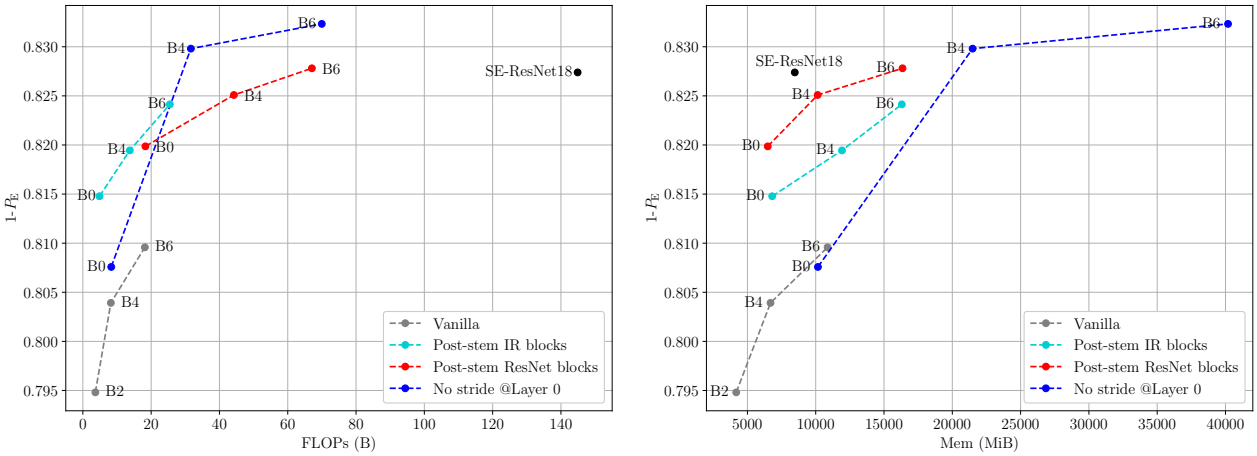

Figure 7: $1 - P_E$ vs. FLOPs and memory requirements of different surgical modifications in the ALASKA II dataset.