A Novel Filter Pruning Algorithm for Vision Tasks based on Kernel Grouping

Jongmin Lee, Armagan Elibol, and Nak Young Chong

Abstract— **Although the size and the computation cost of the state of the art deep learning models are tremendously large, they run without any problem when implemented on computers thanks to the remarkable enhancements and advancements of computers. However, the problem is likely to be faced when the need for deploying them on mobile platforms arises. Model compression techniques such as filter pruning or knowledge distillation help to reduce the size of deep learning models. However the conventional methods contain sorting algorithms therefore they cannot be applied to models that have reshaping layers like involution. In this research, we revisit a model compression algorithm named *Model Diet* that can be both applied to involution and convolution models. Furthermore, we present its application on two different tasks, image segmentation and depth estimation.**

I. INTRODUCTION

Computer vision has been researched since the late 1960s but image classification is still one of the main challenges. After Geoffrey Hinton won the ImageNet Large Scale Vision Recognition Challenge(ILSVRC)[1] as known as ImageNet with AlexNet[2] on 2012, people started researching about Convolutional Neural Networks(CNN) for image classification. Modern neural networks[3], [4] achieved nearly 90% accuracy on the ImageNet dataset, however, the number of parameters is tremendously large. Although the models introduced above have great performance on image tasks, they still require high computational cost, therefore, applying deep neural networks on mobile devices remains challenging.

There were several approaches for reducing the number of parameters while trying not to lose the performance. Knowledge distillation[5] is a method that uses a dense network as a teacher model and a sparse network as a student model. Filter pruning[6] is a method for reducing the number of filters in CNN. When pruning the filters we sort by the sum of weights for each filter for every layer. Since filters that have weights close to 0 will not affect much of the model's performance, when given a proper threshold, we can get a sparse model with similar performance.

However, it is hard to apply the conventional pruning method for involution[7] since it requires sorting operation on filters. Involution has reshaping layers therefore if the filters are sorted, they lose the spatial information. To overcome this problem, we need to rewrite the code for the model which is hard to implement and time-consuming. In this research, we propose a pruning method called the model diet which is easy to implement, and effective for CNN models including involution. Instead of sorting the filters for each layer, we reduce a certain portion of the filters by grouping the kernel weights therefore for involution, the spatial information is not lost. Since the model depth is maintained but the filters are reduced, we call this pruning method a model diet, and we will show that diet models have faster convergence compared with randomly initialized models.

The model diet consists of 2 stages, the kernel grouping stage and the group selection stage. kernel grouping is an algorithm that splits the kernel weights into groups. The kernel weights are split in order therefore when applied to involution, the involution kernel does not lose the spatial information. Once the kernel weights are split into groups, we take the sum of the weights for each group. Then we use the group that has the biggest sum, and we call this operation group selection.

Deep learning frameworks such as TensorFlow or PyTorch save the model's weights as matrices. For involution, the kernel weights are saved as a vector. When the weights are loaded, the vector reshapes itself into the corresponding shape. Therefore, the element of the vector indicates a certain location in an image. If we apply conventional pruning algorithms, the weights of the involution kernel will be sorted also, resulting in a loss of spatial information. However, the kernel grouping keeps the order of the weights, therefore when applied to involution the loss of the spatial information does not happen. Also, the computational complexity of the model diet is $\mathcal{O}(n)$ whereas the computational complexity of the conventional pruning is $\mathcal{O}(n \log n)$. Since model diet has the same computational complexity with selecting the maximum element in a vector where conventional methods have the same computational complexity with sorting.

In this research, we show the effectiveness of the model diet in two computer vision tasks, image segmentation, and depth estimation. We test the performance of the diet model and the randomly initialized model. The diet model showed faster convergence and performance compared with the randomly initialized model. For image segmentation, the dataset was easy to generalize and lacked difficulty therefore the diet model and the randomly initialized model showed equal performance, but the diet model still had faster convergence. For depth estimation, both the diet model and the randomly initialized model showed poor performance even though the loss converged. Also, the difference between groups was studied. We split the full model into 2 groups and compared the performance. Both the group with the bigger sum and smaller sum showed equal performance and speed of convergence. Since pruning can be regarded as weight initialization, we hypothesize that both the group with bigger sums and smaller sums started from a different location but shared the same local optimum point. This work is an extension of our previous work [8] and presents extra analysis on our proposed algorithm and the performance on different vision tasks.

II. Related Work

A. Involution

Involution[7] is a type of kernel that can reduce the inter-channel redundancy of CNNs. It reversed the inheritance of convolutions and has spatial-specific and channel-agnostic features. This term means that Involution kernels refer to the channels for each pixel when generating the kernel weights. Unlike randomly generated kernels like CNN, Involution takes the channel information when generating kernels. This makes a difference because for Involution kernels the kernel weights for each pixel differ. Convolution kernels, on the other hand, share the same weights for every pixel. This makes Involution kernels have wider receptive fields compared to convolution kernels. This makes Involution more similar to self-attention rather than convolution. Self-attention is a relation between 2 pixels but involution takes more pixels in regard, so we can consider Involution a more generalized form of self-attention.

The feature map is defined as $X \in \mathbb{R}^{H \times W \times C_i}$ and a pixel inside the feature map is defined as $X_{i,j} \in \mathbb{R}^{C_i}$. The involution filter is defined as $\mathcal{H}_{i,j} \in \mathbb{R}^{K \times K \times C_i}$. For each layer, the involution kernel exists as the number of output channels therefore each involution layer is defined as $\mathcal{H} \in \mathbb{R}^{H \times W \times K \times K \times C_i}$. The kernel generation function is defined as $\phi : \mathbb{R}^{C_i} \mapsto \mathbb{R}^{K^2 \times C_i}$ where K denotes the involution kernel size. For each pixel $X_{i,j}$, linear projection ϕ will be operated and the output will be reshaped into C_i number of $K \times K$ shaped kernels. Since the involution filter is a concatenated tensor of involution kernels, the entire kernel generation can be written as below.

$$\mathcal{H}_{i,j} = Reshape(\phi(X_{i,j})) \tag{1}$$

Once the involution kernel is generated, we perform a multiply-add operation through the channel dimension. The difference between convolution and Involution is that for convolution, every pixel shares the same kernel. On the other hand, for Involution, the values of the kernel vary depending on the pixels channel therefore all pixels have different kernels.

B. Model compression

Frankle and Carbin [14] hypothesize that there exists a sparse network that has a similar or even better performance compared with the dense network. Although there might exist a winning ticket i.e. the sparse network which has similar performance with the dense network, it is tough to find the initial weights of the parameters. Rather than finding the winning ticket, pruning the weights from a pre-trained model is easier. Knowledge distillation distills the knowledge from a dense model i.e. a teacher model to a sparse one i.e. the student model. Filter pruning prunes the filters of a CNN model and re-trains the pruned model.

Pruning is a commonly used algorithm for model compression. Hao Li et.al. [6] proposed a filter pruning algorithm to reduce the number of weights in CNN. Pruning algorithms like Dropout [11], remove the connections which are also called weights. However, filter pruning prunes the convolution kernel directly instead of pruning the weights. When pruning the kernels, the kernels are sorted by their sum of the weights and a certain proportion among the filters is removed. The reason why the filters are sorted is that the kernels that are closer to 0 are less likely to affect the output.

Zafrir et al. proposed an algorithm named prune once for all (PruneOFA) in [15]. PruneOFA is composed of 2 steps, teacher preparation, and student pruning. The teacher preparation step uses a pre-training dataset to train the teacher model. After training the teacher model, the student model is generated and pre-trained using the teacher model's weights. Then we prune the teacher model using Gradual Magnitude Pruning (GMP) [12] and learning rate rewinding(LRR)[13]. After the pruning is done, we use the task dataset to finetune the pruned model.

Data-free distillation[16] is a knowledge distillation method that does not require data for knowledge distillation. The conventional knowledge distillation method is to minimize the 2 loss functions. One, the error between the ground truth, and second, the error between the teacher model and the student model. However, data-free distillation methods do not require the original data used when training the teacher model. Instead reconstructs the input data by using a random gaussian noise for the teacher model's input. The Gaussian noise goes through the top layer i.e. the classification layer to the bottom layer i.e. the input layer. We use the gradients of each layer to reconstruct the input image.

Data-free adversarial distillation[17](DFAD) is an algorithm that uses a GAN's generator for sample generation. It mimics the learning process of humans. The generator will be trained to sample harder examples. Unlike the previous data-free distillation, this algorithm does not require the layer's activation statistics. Instead, it generates images itself to train the student network.

III. Proposed method

The key point of pruning is that kernels are mutually independent. Therefore it is possible to remove kernels that less affect the performance of the model. However, for algorithms such as involution, deciding which kernel to remove might be difficult. The two main keywords of this research are as follows.

1) Kernel grouping
2) Model Diet

A. Kernel grouping

Involution kernels, the weights are saved as a vector since the kernels act as a linear layer. In order to avoid information loss by the reshaping layer, the order of the kernel weights must be kept. The order of the weights of a deep learning model depends on its shape. For example, convolution layers are saved as (output channels, input channels, K, K) where

TABLE I

NUMBER OF PARAMETERS NEEDED BY THE TYPE OF LAYERS AFTER APPLYING MODEL DIET

Type of layer	Input Size	Output Size	#Parameters($g=2$)	#Parameters($g=3$)
Convolution	$W \times H \times C_i$	$W \times H \times C_o$	$0.25 \cdot (C_i \times C_o \times K^2)$	$0.11 \cdot (C_i \times C_o \times K^2)$
Fully Connected	C_i	C_o	$0.25 \cdot (C_i \times C_o)$	$0.11 \cdot (C_i \times C_o)$
Batch Normalization	$W \times H \times C_i$	$W \times H \times C_o$	$0.5 \cdot (2 \cdot C_o)$	$0.33 \cdot (2 \cdot C_o)$

K stands for the kernel size, and linear projection layers are saved as (output features, input features). The kernel of involution is generated by a linear projection therefore the weights of involution are saved as (output features, input features).

Since reshaping does not contain learnable parameters, the output shape of a reshaping layer is unknown when loading the weights. This means that when the weights of involution are loaded, the weights can be reshaped into an arbitrary shape. To reduce the parameters and keep the kernel weight's order, we need to split kernels into N groups for each layer, where N is the number of the weights. The weights are first split into g groups, where each group G_i takes the index from $(N/g) \times i$ to $(N/g) \times (i+1)$ where i is a natural number smaller than g. We call this operation kernel grouping.

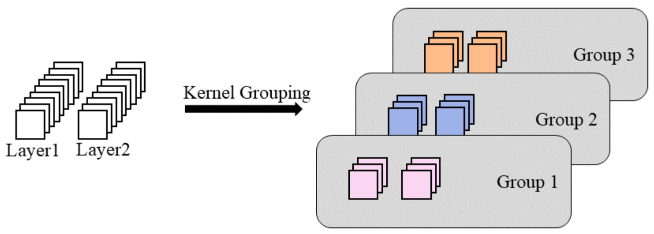

Fig. 1. Kernel Grouping for convolution kernels

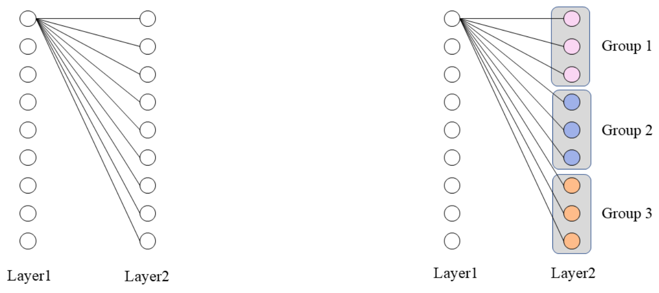

Fig. 2. Kernel Grouping for fully connected layers

Fig. 2 shows kernel grouping for vectors. The figure on the left shows the original model before kernel grouping is applied. The figure on the right shows the weights after kernel grouping is applied. Different groups are colored in differently.

B. Model diet

Fig. 3 shows the visual explanation about the model diet algorithm. The black and white kernels refer to the original model i.e. the full model. Then we group the kernels into N groups. After grouping the kernels we get the total sum of the weights of kernels for each group. Finally, we choose the group that has the biggest sum. We call this model the diet model.

The term *Model diet* comes from reducing the weights while maintaining the model depth. The weights of each group are summed and the group that has the biggest sum is used for the diet model. Our aim is to change the shape of the weight (output channel, input channel, K, K) into (output channel/g, input channel/g, K, K) therefore the weights from index $(N/g) \times i$ to index $(N/g) \times (i+1)$ where i indicates the index of the group which has the max sum of elements are kept.

Fig. 4 shows how the model diet solves the information loss for involution. The vector on the left corresponds to the weights of the kernel generation. The vector on the right indicates the involution kernel after the reshaping layer. Conventional pruning methods need to sort the weights before pruning. That will result in changing the order of the weights and the output kernel will lose the spatial information. On the other hand, the model diet prunes the weights without sorting.

C. Computation Cost

Let K be the width and height of a kernel. Normally a convolution kernel has the same width and height therefore we let the number of weights per kernel be K^2. We define the depth of a model L and the number of kernels per layer as C. Each layer consists of $W \times H \times C$ number of pixels where W and H are the width and height respectively. The size of the convolution kernel for the input layer and the output layer is $C^2 \times K^2$ for the full model. If the model diet is applied the kernel size can be reduced to $(C^2 \times K^2)/g$ since the channel of the input and output layer is unchanged. Note that g is the number of groups. On the other hand, layers in the hidden layer can be reduced to $(C^2 \times K^2)/g^2$. The same logic holds for fully connected layers. The batch normalization layer only depends on the output channels therefore the parameters are reduced to $(C^2 \times K^2)/g$ regardless of the location.

Table I shows the compression rate of the model diet by the type of layers. In most cases, the proportion of the compression stays in g^2. This means, if we choose group the kernels into 2 groups, 75% of the parameters are reduced and 89% if 3 groups.

IV. EXPERIMENTAL RESULTS

A. Compression Rate

We test the compression rate of the model diet on UNet for image segmentation and depth estimation. Table II shows

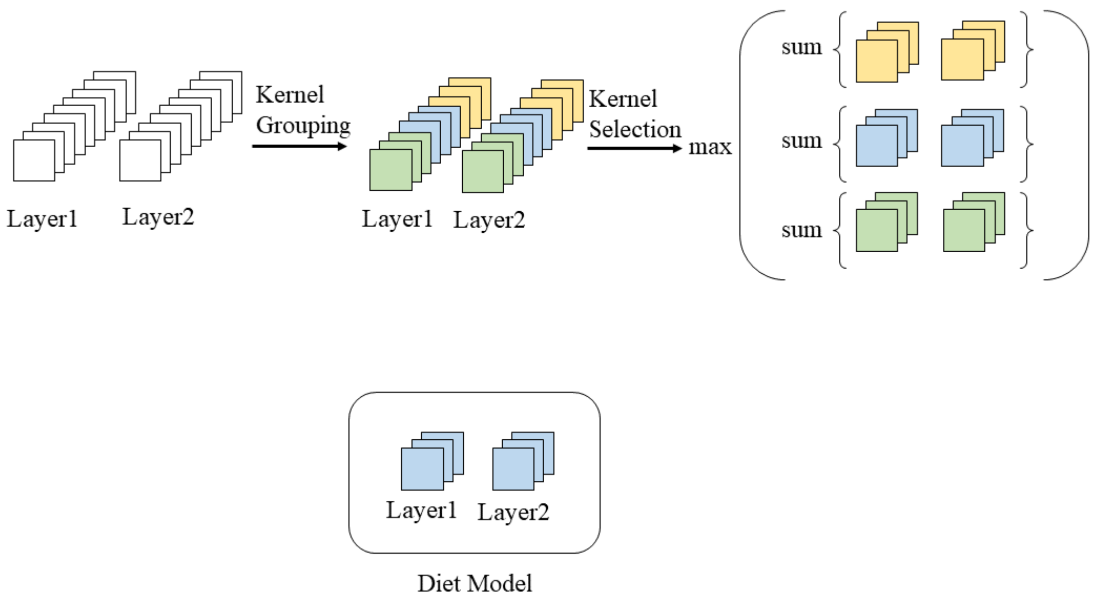

Fig. 3. Visual explanation about model diet algorithm.

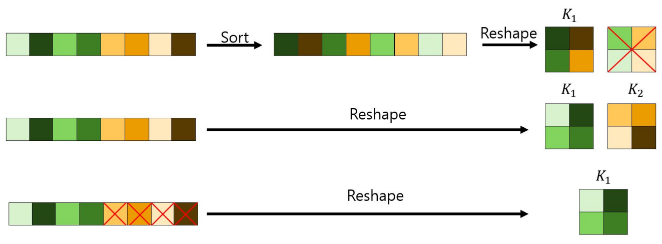

Fig. 4. Comparison between conventional pruning(top) and model diet(bottom). The original involution is showed in the middle

TABLE II
NUMBER OF PARAMETERS ($\times 10^6$) BEFORE AND AFTER DIET.

Model	Full	Diet	Reduction Rate
UNet	17.26	4.32	74.97%

the number of parameters. The diet could reduce up to approximately 75% of the parameters.

Table III shows the number of floating-point operations in a billion scale. It can be seen that the floating-point operations have reduced about 75%. This difference will make a drastic difference in the inference time when it is needed to operate in real time.

TABLE III
NUMBER OF GFLOPS BEFORE AND AFTER DIET.

Model	Full	Diet	Reduction Rate
UNet	10.02	2.52	74.85%

B. *Comparison between Groups*

Fig. 5 and Fig. 6 shows the test accuracy and test loss for the 2 different groups tested with ResNet [9] and VGG [10]. The blue plot indicates the group with the bigger sum of the weights and the orange one indicates the group which used the smaller sum of groups. Either way, the convergence, and performance are similar. Model diet prunes the kernels of the full model which converged into a certain place on the feature space. This means that the optimization leads the weights into a local optimum. For the 2 models, the model which used the bigger sum and the one with the smaller sum will start from a different location in the feature space. Although the 2 models have completely different weights, they share the common convergence point of the full model therefore we hypothesize that the 2 models also share the same convergence point.

C. *Image Segmentation*

The image segmentation task is a task where we classify each pixel in an image. The segmentation model takes an image as an input and outputs a mask. each mask corresponds to a certain label. We test the model diet algorithm on the U-Net model[18]. Other experimental settings are the same with

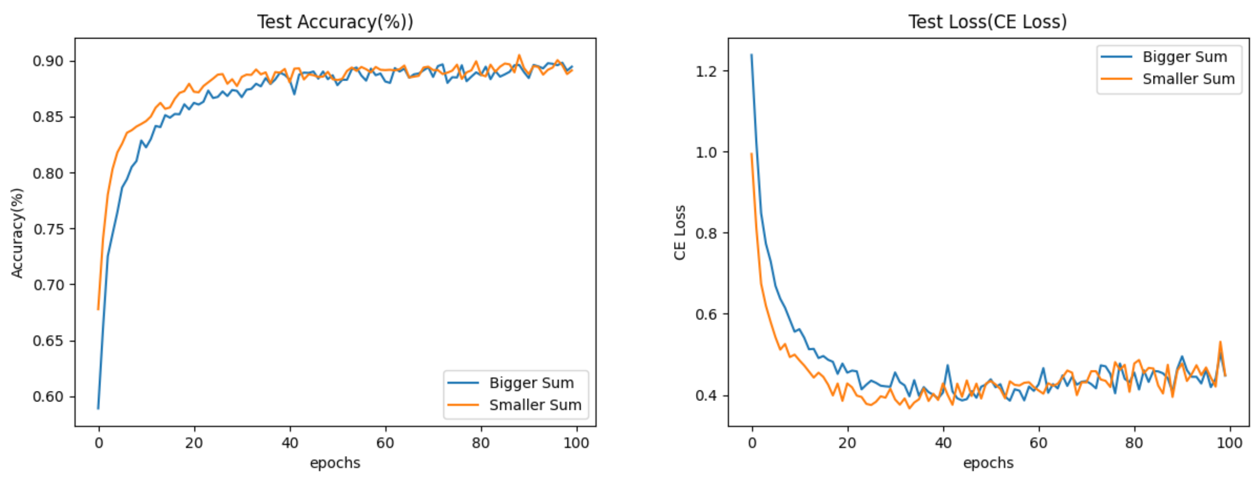

Fig. 5. Comparison between the groups of VGG.

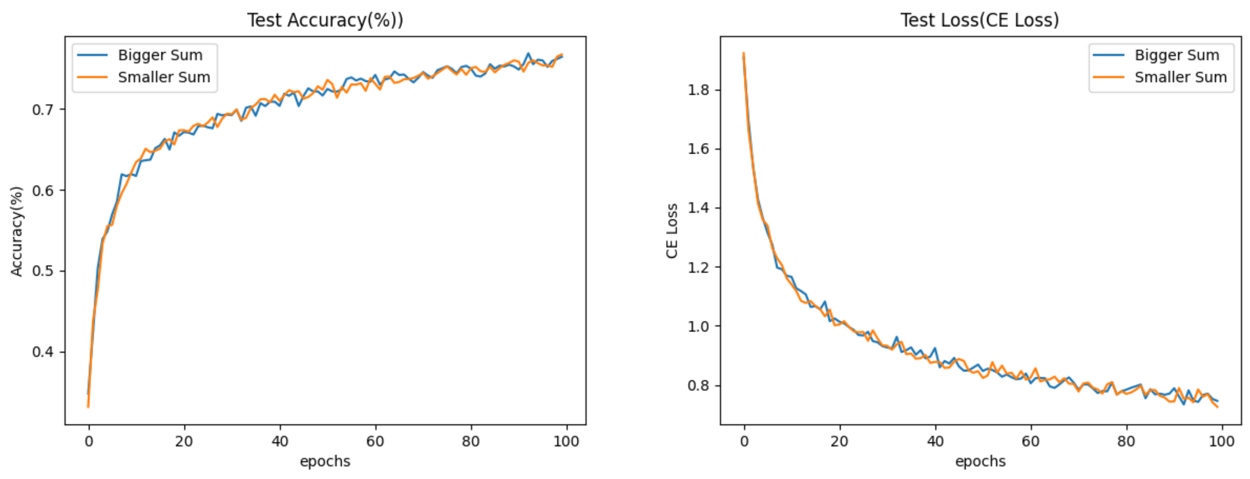

Fig. 6. Comparison between the groups of ResNet

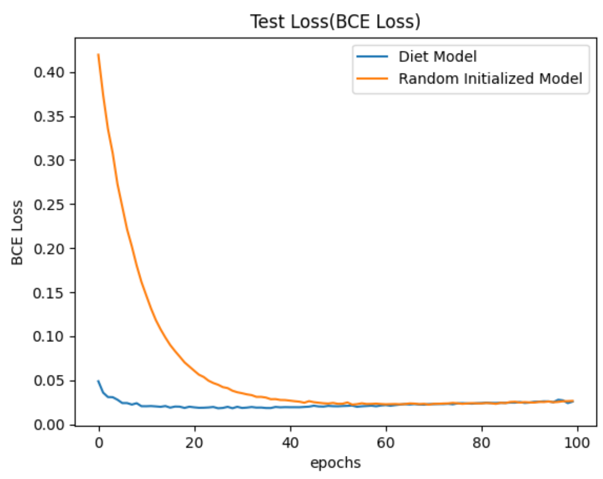

Fig. 7. The test loss of random initialized and the diet model(Tested with U-Net)

image classification. The U-Net model was trained using the Carvana image masking dataset.

Fig. 7 shows the test loss of the diet model and the randomly initialized model of U-Net. We use the binary cross-entropy loss(BCE Loss). The diet model and the randomly initialized model both show similar performance. However, we can still observe that the diet model converges faster than the randomly initialized model. There was no difference in the performance between the diet model and the randomly initialized model since the dataset lacks difficulty. Therefore either model showed similar performance.

D. Depth Estimation

Depth estimation is close to image segmentation, but the classification is changed to regression. We predict the depth for each given pixel in an image. The NYU depth dataset V2 was used for training. Also, the same U-Net model used for image segmentation was used for depth estimation. The training settings stay the same with image classification.

Fig.8 shows the test loss(MSE loss) for the diet model and the randomly initialized model. Both diet model and ran-

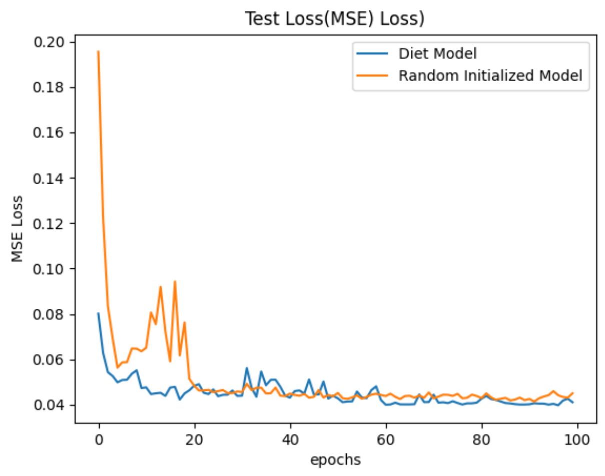

Fig. 8. The test loss of random initialized and the diet model for depth estimation(Tested with U-Net)

domly initialized model showed convergence around epoch 20. The performance of the diet model is slightly higher than the randomly initialized model. Although the loss was reduced, the prediction of the U-Net model was poor. Since we used MSE loss for training, and MSE loss does not guarantee the local information the decrease of the MSE loss does not mean that the performance is getting better.

V. CONCLUSIONS

We tested the performance and convergence on 2 other tasks. For image classification, the diet model showed better performance and faster convergence compared to the randomly initialized model. For image segmentation, both the diet model and the randomly initialized model showed similar performance. However, the diet model still had faster convergence compared to the randomly initialized model. For depth estimation, we used the same model architecture with image segmentation. The performance was poor even though the loss converged. The summation of groups did not affect much on the performance. We hypothesize that both groups with the bigger sum and the smaller sum exist in the same local optimum.

Both the diet model and the randomly initialized model trained for image segmentation had similar performance. This happens due to the lack of difficulty of the dataset. To test the effectiveness of the model diet, the U-Net model must be tested on harder datasets such as the COCO image segmentation dataset.

We failed to train the U-Net model for depth estimation. To check whether the model diet works on depth estimation, we must try state-of-the-art depth estimation models or other training methods. In this research monocular depth estimation was tested with U-Net with MSE loss trained with the Adam optimizer. For future work, other state-of-the-art models and training methods must be tested.

For future work, some experimental and performance analyses will be targeted in other models for object detection, image generation, and similar other tasks. Also, more sophisticated ways to select a group of weights apart from the summation of weights and the effect of the sum of the groups will be studied. To check if the weights are located into a similar local optimum, similarity functions such as cosine similarity will be used to compare the values of the model's weights. If the 2 models have big similarities, we can conclude that 2 models converged into the same local optimum. Also, if the models share the same convergence point, it means that we can select either group and therefore the algorithm can be simplified.

REFERENCES

[1] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In:2009 IEEE conference on computer vision and pattern recognition. Ieee. 2009,pp. 248–255.
[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classi-fication with Deep Convolutional Neural Networks”. In:Advances in NeuralInformation Processing Systems. Ed. by F. Pereira et al. Vol. 25. CurranAssociates, Inc., 2012.
[3] Hieu Pham et al. “Meta Pseudo Labels”. In:CoRRabs/2003.10580 (2020).arXiv:2003.10580.url:https://arxiv.org/abs/2003.10580.
[4] Andrew Brock et al. “High-Performance Large-Scale Image Recognition With-out Normalization”. In:CoRRabs/2102.06171 (2021). arXiv:2102.06171.url:https://arxiv.org/abs/2102.06171.
[5] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean.Distilling the Knowledge ina Neural Network. 2015. arXiv:1503.02531 [stat.ML].
[6] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet,and Hans Peter Graf.Pruning Filters for EfficientConvNets. 2017. arXiv:1608.08710 [cs.CV].
[7] Duo Li et al.Involution: Inverting the Inherence of Convolution for VisualRecognition. 2021. arXiv:2103.06255 [cs.CV].
[8] J. Lee, A. Elibol and N. Y. Chong, ”Model Diet: A Simple yet Effective Model Compression for Vision Tasks,” 2021 21st International Conference on Control, Automation and Systems (ICCAS), 2021, pp. 506-511, doi: 10.23919/ICCAS52745.2021.9649988.
[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, andJian Sun. “Deep Residual Learning for ImageRecognition”. In:Proceedings of the IEEE Confer-ence on Computer Vision and Pattern Recognition(CVPR). June 2016.
[10] Karen Simonyan and Andrew Zisserman.VeryDeep Convolutional Networks for Large-ScaleImage Recognition. 2015. arXiv:1409 . 1556[cs.CV].
[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky,Ilya Sutskever, and Ruslan Salakhutdinov. “Dropout:A Simple Way to Prevent Neural Networks fromOverfitting”. In:Journal of Machine LearningResearch15.56 (2014), pp. 1929–1958.URL:http : / / jmlr . org / papers / v15 /srivastava14a.html
[12] Michael Zhu and Suyog Gupta.To prune, or notto prune: exploring the efficacy of pruning formodel compression. 2017. arXiv:1710.01878[stat.ML].
[13] Alex Renda, Jonathan Frankle, and MichaelCarbin. “Comparing Rewinding and Fine-tuningin Neural Network Pruning”. In:CoRRabs/2003.02389(2020). arXiv:2003.02389.
[14] Jonathan Frankle and Michael Carbin.The LotteryTicket Hypothesis: Finding Sparse, Trainable NeuralNetworks. 2019. arXiv:1803.03635 [cs.LG]
[15] Ofir Zafrir, Ariel Larey, Guy Boudoukh, HaihaoShen, and Moshe Wasserblat. “Prune Once for All:Sparse Pre-Trained Language Models”. In:CoRRabs/2111.05754 (2021). arXiv:2111.05754.URL:https://arxiv.org/abs/2111.05754.
[16] Raphael Gontijo Lopes, Stefano Fenu, and Thad-Starner. “Data-Free Knowledge Distillation for DeepNeural Networks”. In:CoRRabs/1710.07535 (2017).arXiv:1710.07535.URL:http://arxiv.org/abs/1710.07535.
[17] Gongfan Fang, Jie Song, Chengchao Shen, Xin-chaoWang, Da Chen, and Mingli Song. “Data-Free Adver-sarial Distillation”. In:CoRRabs/1912.11006 (2019).arXiv:1912.11006.URL:http://arxiv.org/abs/1912.11006.
[18] Olaf Ronneberger, Philipp Fischer, and ThomasBrox.U-Net: Convolutional Networks for Biomed-ical Image Segmentation. 2015. arXiv:1505 .04597 [cs.CV].