Analysing Efficiency of Deep Multi-Task Networks Using Distinctiveness Pruning

Jaskirat Singh[1]

Australian National University
Research School of Computer Science
Canberra, Australia
u7019589@anu.edu.au

Abstract. Multi-task learning refers to using a single encoder network in order to learn multiple diverse tasks over the same dataset. The use of common features for multiple different tasks is not only helpful in improving the generalization performance of the final model but also holds high significance in practical deployment of such multi-task networks while using limited disk space. The current research focused on reducing the size of multi-task models often incorporates use of techniques like neural architecture search (NAS) in order to find the smallest possible network size. In this paper, we instead analyse the merits of using the distinctiveness pruning strategy [3] for network size compression in deep multi-task models, and show that it leads to consistent performance improvements as compared to directly training a network of final compressed size. Furthermore, we note that since the same encoded features are used for simultaneous predictions on multiple diverse tasks, the size of the learned feature representation forms a major bottleneck for both resulting performance and the final model size. To this end, we propose a novel distinctiveness pruning based approach to analyse the effect of using different convolution architectures (*e.g.* ResNet [4], Efficient-Net [12], NF-Net [1]) on the effective utilization of the final encoded feature representation. Our results show that smaller networks like Efficient-Net perform on-par with bigger networks by exhibiting increased effective utilization of the joint encoded features. Source code is available at https://gitlab.cecs.anu.edu.au/u7019589/img-compression-vehiclex.

Keywords: Multi-task learning · Network Compression · Pruning · Vehicle-X

1 Introduction

The deployment of current deep learning models for practical computer vision applications, *e.g.* in robotics or self-driving cars, requires the final network to solve multiple diverse tasks at once. For instance, a self-driving car needs to simultaneously perform a range of visual tasks like semantic segmentation (for detecting different pedestrians and other vehicles), per-pixel depth estimation (for avoiding obstacles) and visual navigation (for optimal path planning) *etc*, all within a few-milliseconds of time. A typical way to reduce the computation required at inference time is to use a single network to solve multiple tasks at the same time. This approach, also known as *multi-task learning*, is not only helpful in reducing inference time, but also offers additional benefits in terms of improved generalization performance and the need for limited network size (which is important for deployment in mobile systems with limited space).

A key requirement in multi-task learning is to design a common *encoder* network (Fig. 1) which solves multiple tasks while using limited disk space. The current work in this direction, often uses techniques like neural architecture search (NAS) in order to find the most efficient neural network design for the encoder model [2,7,8]. While effective, neural architecture search (NAS) techniques require multiple training runs in order to perform task-agnostic search across possible network design parameters. The obvious disadvantage of such an approach can be seen in terms of its high training time, which highlights the need for developing on-line / progressive techniques for reducing network size. To this end, we

analyse the merits of using the distinctiveness pruning [3] technique in order to perform progressive compression of the final deep multi-task model based on the online training dynamics.

Another way to reduce network size is to use a smaller but efficient encoder architecture for deep multi-task models. For instance, recent years have seen the advent of several efficient convolutional architectures like ResNet [4], EfficientNet [8,13], ShuffleNet [6], NFNet [1] *etc*, which exhibit increasingly smaller network size without compromising on the final model performance. However, the current literature lacks a unified approach to explain the increased efficiency of these networks and the results are often attributed to a large-scale empirical analysis. In this paper, we use distinctiveness pruning in order to propose a novel approach for explaining the performance of these networks. In particular, we show that increasingly smaller networks like Efficient-Net [12] perform on-par with bigger networks (with higher parameter count) by exhibiting increased utilization of the encoded feature representations.

To summarize the main contributions of this paper are as follows,

1. Demonstrate and analyse the effectiveness of distinctiveness pruning approach for progressive network compression in multi-task learning.
2. Show that the incorporation of each additional task in multi-task learning leads the network to learn increasingly diverse / non-repetitive features which increases the efficiency of the final model.
3. Propose a novel distinctiveness pruning based approach for explaining the performance of smaller but efficient convolution architectures (*e.g.* EfficientNet [8,13], Xception [6], ShuffleNet [6]).
4. Show that increasingly smaller networks like Efficient-Net perform on-par with bigger networks (with higher parameter count) by exhibiting increased utilization of the learned features.

2 Our Method: Progressive Network Compression in Multi-task Learning

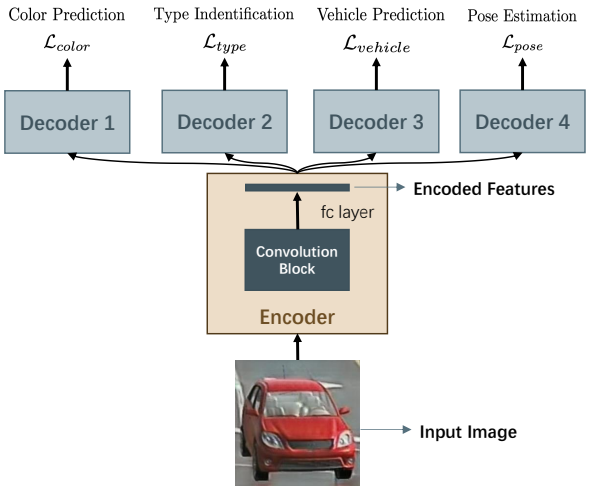

Fig. 1. Overall Model Design for Deep Multi-task Learning.

In this section, we provide an outline for our overall approach on applying distinctiveness pruning to multi-task models on the Vehicle-X dataset. We begin with an introduction of the overall multi-task learning setup in Section 2.1, followed by a discussion on using hard sample mining as a means to deal with unbalanced class data (refer Section 2.2). We then discuss our approach for performing distinctiveness pruning in Section 2.3. Finally, we use gradient surgery / PCGrad [15] to remove negative inter-task interference while performing network compression in multi-task learning (refer Sec. 2.4).

2.1 Overall Multi-task Learning Setup

We follow the traditional encoder-decoder architecture for multi-task learning [16]. The encoder network is composed of a convolutional feature extraction network (*e.g.* ResNet-18) followed by a single fully connected layer which learns a joint hidden representation $H \in \mathbb{R}^h$. The joint hidden representation is then fed into multiple decoder networks to form a prediction for each of the output tasks. For the purpose of this paper, all decoder networks are composed of a single fully connected layer which maps the hidden feature ($H \in \mathbb{R}^h$) to the output prediction ($Y_i \in \mathbb{R}^{k_i}, i = 1, 2...n$) for each task. In above, k_i is the output dimension for each task and n is the total number of tasks. Fig. 1 provides on overview of the multi-task model design.

The final model is then trained end-to-end using a weighted combination of each of the task-specific loss functions [16]. In particular for the Vehicle-X dataset [14], we perform multi-task learning over four diverse classification tasks: color prediction, vehicle type identification, vehicle ID prediction and pose estimation. The task of pose estimation is in general not a classification task. However, it is possible to make such an equivalence for the Vehicle-X dataset, as we found that each image was captured from a finite set of global camera poses. Thus the pose estimation task is posed as predicting the camera ID of the camera from which a particular image was captured. Mathematically, we define the overall loss function as,

$$\mathcal{L}_{multi-task} = \lambda_{color}\mathcal{L}_{color} + \lambda_{type}\mathcal{L}_{type} + \lambda_{vehicle}\mathcal{L}_{vehicle} + \lambda_{pose}\mathcal{L}_{pose} \tag{1}$$

where $\{\mathcal{L}_{color}, \mathcal{L}_{type}, \mathcal{L}_{vehicle}, \mathcal{L}_{pose}\}$ and $\{\lambda_{color}, \lambda_{type}, \lambda_{vehicle}, \lambda_{pose}\}$ are the cross-entropy based loss functions and weighting factors, respectively, for each of the prediction tasks.

2.2 Hard Sample Mining for Dealing with Unbalanced Class Data

An important challenge while performing multi-task learning on the Vehicle-X dataset is to deal with the problem of unbalanced class sizes for different tasks. Now, a trivial way for dealing with class imbalances in single task learning would be to adjust the weightage / sampling probability for each class to be inversely proportional to the the overall class size. However, the same is not directly feasible in multi-task learning, as we observe that each task can correspond to a different class size distribution. For instance, we observed that the vehicle color class *pink* was found to only occupy $\sim 2.26\%$ of the overall training data. However, simply increasing the weightage / sampling probability of images with *pink* colored vehicles would implicitly imbalance the class distribution for other tasks, like vehicle type identification. This is because the conditional distribution $P(type \mid color = pink)$ is another highly non-uniform distribution with probabilities,

$$P(type \mid color = pink) \approx [0.35, 0.21, 0.09, 0.24, 0, 0, 0, 0.11, 0, 0, 0] \tag{2}$$

Thus, we see that performing class balancing for *color prediction task* by increasing the weightage / sampling probability of images with *pink* colored vehicles would implicitly unbalance the data against vehicles of type *sportscar* which corresponds to tenth class for *vehicle type identification task*.

Hard sample mining. To address the above problem, we use a hard sample mining approach which adaptively increases the weight-age of samples which are currently hard for the multi-task network. To do this, given a batch of input data X, we rank the multi-task loss from Eq. 1 for each sample in a descending order. The final loss is then computed as the mean over the sample-specific losses for the hardest 70% of the batch samples.

2.3 Distinctiveness Pruning

Feature learning in neural networks can sometimes be quite redundant, often leading to learning of similar or opposing features. To address this, we incorporate the distinctiveness pruning approach

[3] which removes hidden nodes from the encoder output layer based on the distinctiveness of their output / activation patterns on the training data. In this section, we propose a generalized variation of distinctiveness pruning proposed in [3], which allows for pruning similar / dissimilar features nodes while ensuring a more robust behaviour to similarity thresholds (θ_{thresh}).

Given the batch of input data X, we first pass it through the convolutional feature extractor network to get the features $X_f \in \mathbb{R}^{N \times d}$. The hidden state output activation patterns for all h hidden nodes are then computed as,

$$H = \{\mathbf{h}_1, \mathbf{h}_2, \dots \mathbf{h}_h\}, \quad \mathbf{h}_i \in \mathbb{R}^N \ \forall i = \{1, 2 \dots k\} \tag{3}$$

$$\mathbf{h}_i = \sigma\left(X_f \ \mathbf{w}_i^{(1)}\right), \quad \mathbf{w}_i^{(1)} \in \mathbb{R}^d \tag{4}$$

where $\mathbf{w}_i^{(1)} \in \mathbb{R}^d$ is the weight vector for the fully connected encoder layer corresponding to the i^{th} hidden output, and $\sigma(.)$ is the activation function.

We can then compute the similarity S_{ij} between the output patterns for pair of hidden nodes i, j as,

$$S_{ij} = \cos\left(\angle(\mathbf{h}_i, \mathbf{h}_j)\right) = \frac{\mathbf{h}_i \ . \ \mathbf{h}_j}{\|\mathbf{h}_i\| \ \|\mathbf{h}_j\|} \tag{5}$$

Core Idea. The main idea behind distinctiveness pruning is to use the hidden activation pattern similarities (computed above) to remove a hidden node while ensuring similar overall *output* prediction pattern for all output nodes on each task. To see this, let $\mathbf{o}_p^m \in \mathbb{R}^N, p \in \{1, 2, \dots k_m\}$ be the *output* prediction pattern for the p^{th} output node for the m^{th} task. Then,

$$\mathbf{o}_p^m = \sum_{i=1}^{h} w_{pi}^{(2)m} \ \ \mathbf{h}_i \tag{6}$$

where $w_{pi}^{(2)m}$ is the weight between i^{th} hidden node to p^{th} output node in the decoder layer for the m^{th} task.

Now given two hidden nodes x, y with similarity $|S_{xy}| = |\cos\left(\angle(\mathbf{h}_x, \mathbf{h}_y)\right)| \approx 1$,

$$\mathbf{h}_y \approx \mathbf{h}_x \cos\left(\angle(\mathbf{h}_x, \mathbf{h}_y)\right), \quad \text{assuming} \, |\mathbf{h}_x| = |\mathbf{h}_y| \tag{7}$$

We can then write the expression for output activation pattern $\mathbf{o}_p^m$ as,

$$\mathbf{o}_p^m = \sum_{i \neq \{x,y\}}^{h} w_{pi}^{(2)m} \ \ \mathbf{h}_i + w_{px}^{(2)m} \mathbf{h}_x + w_{py}^{(2)m} \mathbf{h}_y \tag{8}$$

$$= \sum_{i \neq \{x,y\}}^{h} w_{pi}^{(2)m} \ \ \mathbf{h}_i + w_{px}^{(2)m} \mathbf{h}_x + w_{py}^{(2)m} \cos\left(\angle(\mathbf{h}_x, \mathbf{h}_y)\right) \mathbf{h}_x \tag{9}$$

$$= \sum_{i \neq \{x,y\}}^{h} w_{pi}^{(2)m} \ \ \mathbf{h}_i + \left(w_{px}^{(2)m} + w_{py}^{(2)m} \cos\left(\angle(\mathbf{h}_x, \mathbf{h}_y)\right)\right) \mathbf{h}_x \tag{10}$$

$$= \sum_{i=1}^{h-1} w_{pi}^{(2)m''} \ \ \mathbf{h}_i \tag{11}$$

where,

$$w_{pi}^{(2)m''} = \begin{cases} w_{px}^{(2)m} + w_{py}^{(2)m} \cos\left(\angle(\mathbf{h}_x, \mathbf{h}_y)\right) & \text{if} \quad \text{i=x} \\ w_{pi}^{(2)m} & \text{otherwise} \end{cases} \tag{12}$$

Thus, we can essentially remove the hidden node y and replace $w_{px}^{(2)m}$ with $w_{px}^{(2)m} + w_{py}^{(2)m} \cos\left(\angle(\mathbf{h}_x, \mathbf{h}_y)\right)$, while preserving the overall output pattern behaviour for the pruned network.

We now use the above idea in order to discuss two different pruning strategies based on similarity / dissimilarity of the hidden pattern activations.

Similarity Pruning. Similarity pruning is performed when two hidden nodes x, y have pattern activations in similar directions *i.e.* $\cos\left(\angle(\mathbf{h}_x, \mathbf{h}_y)\right) \approx 1$ or $\angle(\mathbf{h}_x, \mathbf{h}_y) \in [0, \theta_{threshold}]$. Thus using Eq. 12, we can remove the hidden node y while replacing $w_{px}^{(2)m}$ with,

$$w_{px}^{(2)m} \leftarrow w_{px}^{(2)m} + w_{py}^{(2)m} \cos\left(\angle(\mathbf{h}_x, \mathbf{h}_y)\right) \qquad \forall p \; \forall m \tag{13}$$

Note that for $\cos\left(\angle(\mathbf{h}_x, \mathbf{h}_y)\right) \approx 1$, this directly corresponds to the additive weight update rule $w_{px}^{(2)m} \leftarrow w_{px}^{(2)m} + w_{py}^{(2)m}$ from [3].

Dissimilarity Pruning. Similary, if two hidden nodes x, y have pattern activations in opposing directions *i.e.* $\cos\left(\angle(\mathbf{h}_x, \mathbf{h}_y)\right) \approx -1$ or $\angle(\mathbf{h}_x, \mathbf{h}_y) \in [180 - \theta_{threshold}, 180]$. Thus using Eq. 12, we can remove the hidden node y while replacing $w_{px}^{(2)m}$ with,

$$w_{px}^{(2)m} \leftarrow w_{px}^{(2)m} + w_{py}^{(2)m} \cos\left(\angle(\mathbf{h}_x, \mathbf{h}_y)\right) \qquad \forall p \; \forall m \tag{14}$$

However for $\cos\left(\angle(\mathbf{h}_x, \mathbf{h}_y)\right) \approx -1$, this correpsonds to,

$$w_{px}^{(2)m} \leftarrow w_{px}^{(2)m} - w_{py}^{(2)m} \qquad \forall p \; \forall m \tag{15}$$

Note: in our experiments, we found the above rule for dissimilarity pruning to give much more robust performance as compared to directly removing both nodes x, y.

2.4 Avoiding Negative Inter-Task Transference: PCGrad

A typical phenomenon in deep multi-task learning occurs in the form of *negative transfer* across tasks. This usually happens when the increase in performance of one task causes a performance decrease for the other. This usually occurs due to use of a small network size for the multi-task model and is often accompanied by learning of opposing features for different tasks [11]. However, this directly presents a problem for the dissimilarity pruning strategy outlined in Sec. 2.3, which is based on removal of hidden nodes with opposing pattern behaviour. This can become highly undesirable especially if the reduction in network size from pruning leads to an increased *negative transfer* across tasks.

To avoid this problem while performing distinctiveness pruning on multi-task networks, we adopt the gradient surgery / PCGrad [15] approach for avoiding opposing task gradients in multi-task learning. Given two opposing gradients $\mathbf{g}_i, \mathbf{g}_j$ from task i, j, PCGrad updates the gradients as follows [15],

$$\mathbf{g}_i^{PC} = \mathbf{g}_i - \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{\|\mathbf{g}_j\|^2} \, \mathbf{g}_j, \quad \text{if} \quad \mathbf{g}_i \cdot \mathbf{g}_j < 0 \tag{16}$$

3 Dataset Analysis

The final progressive network compression framework described in Sec. 2, is deeply based on the statistical analysis of the underlying dataset (*e.g.* Section 2.2 for countering class imbalance).

3.1 Dataset Description and Task Identification for Multi-task Learning

The vehicle-X dataset is a high-fidelty dataset consisting of total 75516 images (each provided as a 256 × 256 RGB image) and is divided into train/val/test sets of sizes 45438, 14936 and 15142 respectively. Each data point is accompanied by a set of 9 different labels, out of which 4 are categorical: (colorID, typeID, vehicleID and cameraID), and the rest are continuous regression variables for camera height, camera distance, lighting conditions etc.

For the purpose of this paper, we use the above mentioned four categorical variables in order to analyse the performance of distinctiveness pruning for reducing the size of joint-encoder based deep multi-task networks. Note that, all "string" based categorical variables were converted to their equivalent numeric labels *e.g.* cameraID 'c001' → 1, cameraID 'c005' → 5 *etc.*

Complexity of each task in joint multi-task learning. The overall muti-task network is designed to be of sufficiently high complexity as the final model has to use the same highly pruned feature representation, in order to make simultaneous predictions on several diverse tasks. Another factor affecting complexity of the involved tasks comes from fine-grain classification requirements for each task. For instance, the vehicle identification task requires fine-grain classification into one of 1362 classes, which poses a steep challenge since we only have limited (∼ 30) training images for each vehicle ID.

Note on representation for colorID. We note that the given data classifies each color as a separate categorical variable, which does not capture the visual similarity between different color classes *e.g.* yellow is closer to golden than red. Thus, ideally the target label for each color should be a continuous RGB representation and the color prediction task should be modelled as a regression task. Nonetheless, we found that modelling it as a classification objective helps us evaluate performance on this task with an accuracy metric, which preserves consistent evaluation of the final multi-task network in terms of the overall average accuracy (refer Table 1).

3.2 Highlighting Class Imbalance

A key problem with performing multi-task learning on the Vehicle-X dataset can be seen in terms of its highly imbalanced and distinct class size distributions for different tasks. As discussed in Sec. 2.2, unlike single task-learning, such an imbalance is hard to address with traditional loss re-weighting strategies.

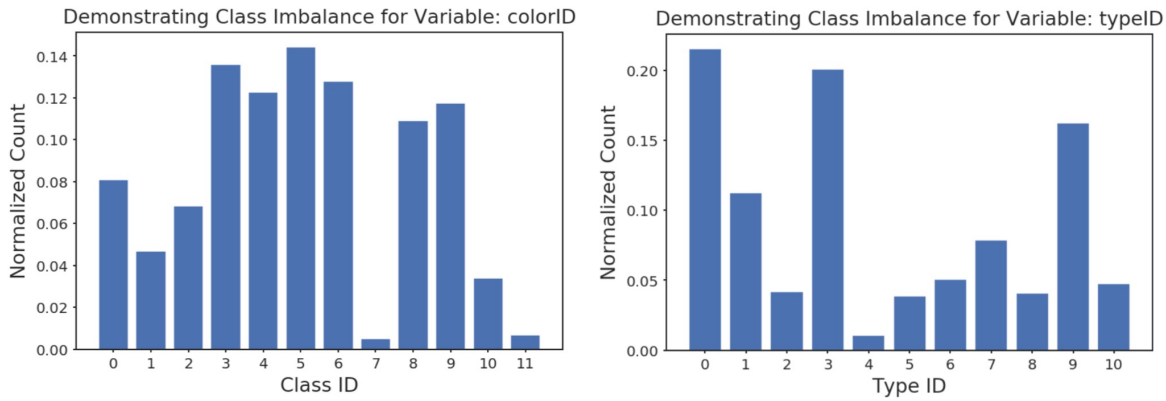

Fig. 2. Highlighting class imbalance and need for hard sample mining. As shown above we find that the given vehicleX dataset features highly class imbalance for different classification tasks. This imbalance poses a direct challenge to the overall multi-task learning process, which highlights the need for the hard sample mining strategy proposed in Section 2.

4 Training Details

Tasks. We train the deep multi-task model on four classification tasks on the Vehicle-X dataset: color prediction, vehicle type identification, vehicle ID prediction and pose estimation. As mentioned in Section 2.1, the pose prediction problem was adapted as a classification task by predicting the camera ID of the camera from which a given image was captured.

Finetuning. In order to increase the speed of the training progress, we follow a finetuning procedure where the weights of the convolutional feature block are initialized from the weights pretrained on the Imagenet [5] dataset. Nevertheless we note that the weights of the convolutional feature block are not kept fixed and the entire multi-task model is trained end-to-end on the Vehicle-X [14] dataset.

Preprocessing / Data Augmentation. We follow the standard image normalization procedure,

$$X \leftarrow \frac{X - \mu}{\sigma} \tag{17}$$

where $\mu = [0.485, 0.456, 0.406]$, $\sigma = [0.229, 0.224, 0.225]$ are the mean and standard deviation on the Imagenet training dataset. This is required due to the pretrained nature of network weights.

Furthermore, in order to increase the diversity of the input data distribution we perform the random horizontal flipping and random crop based image transformations on the training dataset.

Regularization. The high parameter count of the overall deep multi-task model, makes it highly susceptible to overfitting on the training dataset. To address this, we did two things: 1) We added a batch normalization layer after the final fully connected layer in the encoder network, and 2) we added an l_2 regularization term ($\lambda_{reg} = 5 \times 10^{-3}$) to the final overall network loss.

Hyperparameters / Pruning schedule. Unless otherwise specified, each training configuration is ran for 20 epochs with a starting learning rate of 10^{-3}. We use the Adam optimizer with an adaptive learning rate schedule, where the learning rate is reduced by factor of 10 if sufficient progress is not made over the last few epochs. The network is initialized with 128 hidden nodes for the fully connected encoder layer. Pruning is performed in an iterative fashion in conjunction with network training at the end of each epoch. Note that, distinctiveness pruning is only performed if their is a pair of similar / dissimilar neurons within a threshold $\theta_{thresh} = 30$ degrees.

5 Experiments and Results

5.1 Using Distinctiveness Pruning for Network Compression.

In this section, we demonstrate the effectiveness of using the distinctiveness pruning approach for progressive network compression in deep multi-task learning. We use the following training configurations for this experiment,

Distinctiveness Pruning (D-Prune). The network is initialized with 128 hidden neurons for the final encoder layer. The distinctiveness pruning strategy outlined in Sec. 2.3 is then used to iteratively reduce the number of hidden nodes after every epochs using $\theta_{thresh} = 30$ degrees.

Baseline. Let the final number of hidden neurons (in the final encoder layer) after distinctiveness pruning be x. We then train a vanilla deep multi-task network (without pruning) with x hidden nodes in the encoder layer. The network is trained for 20 epochs and the final accuracy across different tasks is compared with the performance after distinctiveness pruning.

Note that both training configurations use a ResNet-50 [4] architecture (without the final fully connected layers) as the convolution feature extractor block for the encoder network. The convolution layer weights are initialized from the pretrained Imagenet model, but are then finetuned on the Vehicle-X dataset.

Multi-Task Performance with Resnet-50 Backbone		
Task	D-Prune	Baseline
ColorID	**90.51**	89.88
TypeID	**76.01**	72.98
VehicleID	**56.44**	52.89
Average	**74.05**	71.92

Table 1. Distinctiveness pruning for network compression. We clearly see that our framework (refer Sec. 2) for applying distinctiveness pruning achieves higher test accuracy (for similar network compression) as compared with the baseline model for both single / multi-task learning.

5.2 Analysing Impact of Network Architecture on Effective Feature Utilization

In this section, we analyse the impact of using different network architectures for convolution block in the encoder network. In particular, we use network compression-ability (using distinctiveness pruning with constant threshold $\theta_{thresh} = 30$)) to explain the competitive performance of smaller but efficient network designs (*e.g.* EfficientNet [12]). Our results indicate that the use of more efficient network architectures increases the effective utilization of the jointly learned features by the encoder network.

Convolution Block Architecture	Parameter Count (M)	Initial Hidden Size	Final Hidden Size	Compression Performance (%)	Feature Utilization (%)
EfficientNet-B0 [12]	5.29	128	109	14.84	85.16
EfficientNet-B2 [12]	9.11	128	107	16.41	83.59
ResNet-18 [4]	11.69	128	102	20.31	79.69
ResNet-50 [4]	25.56	128	93	27.33	72.67
NFNet-L0 [1]	35.07	128	80	37.51	62.49
VGG-11 [9]	132.87	128	65	46.85	53.15

Table 2. Analysing impact of different network architectures on the effective feature utilization. We clearly see that smaller (but efficient) network architectures like EfficientNet perform on par with bigger networks like VGG-11 by increasing the effective utilization of the encoded feature representation.

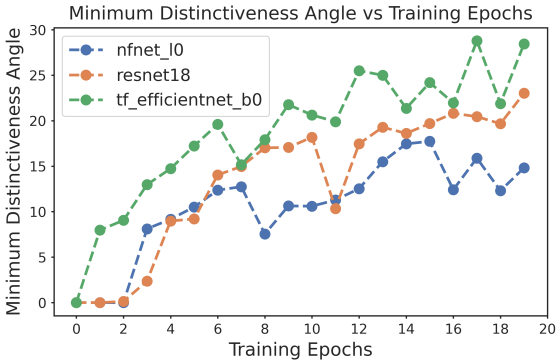

Fig. 3. Variation of minimum distinctiveness angle with training epochs We clearly see that the learned features become increasingly diverse as the training progresses. We also observe that the feature distinctiveness is higher for more efficient network architectures with high feature utilization (refer Table 2).

5.3 Analysing Relation between Compression Performance and Number of Tasks

A key claimed advantage of multi-task learning is its ability to increase the network data efficiency by maximizing the utilization of the learned feature space. In this section, we empirically verify the validity of the above claim by analysing the variation in maximum compression ability (using distinctiveness pruning with constant threshold $\theta_{thresh} = 30$) as the number of tasks is increased.

Multi-task Performance with Resnet-50 Backbone				
# of Tasks	Initial Hidden Size	Final Hidden Size	Compression Performance (%)	Equivalent Feature Utilization (%)
1	128	76	40.63	59.37
2	128	83	35.16	64.84
3	128	93	27.33	72.67
4	128	101	21.09	78.91

Table 3. Analysing Relation between Compression Performance and Number of Tasks. We clearly see that an increase in number of tasks is followed by a reduction in the compress-ability of the original network, which implies that multi-task learning helps in increasing utilization of the given feature space.

6 Conclusion and Future Work

In this paper, we analysed the merits of adopting a distinctiveness based pruning approach for reducing network size in deep multi-task learning models. Furthermore, with the help of appropriate dataset analysis, we highlighted the problem of class imbalance in multi-task learning and proposed a hard sample mining based solution for dealing with the problem. We also discussed how the problem of negative transfer among different tasks can pose a challenge in applying dissimilarity pruning for joint-encoder based deep multi-task networks. This analysis then motivated us to use PCGrad based gradient surgery [15] in order to reduce opposing gradients among different tasks, thus limiting the problem of negative transfer.

Next, we proposed a generalized extension of the distinctiveness pruning approach based on extensive mathematical analysis presented in Sec. 2.3. The proposed approach attempts to remove hidden nodes from the final encoder layer while preserving the overall output activation pattern for each of the tasks, and helped us identify a robust extension for applying distinctiveness pruning on pairs of hidden nodes with opposing feature patterns. Finally, through extensive quantitative evaluation on the VehicleX dataset we demonstrated the merits of using distinctiveness based node pruning for progressive network compression. We also found that the maximum compression performance on a multi-task network decreases as the number of tasks is increased. This result along with analysis from Sec. 5.3 helped us realize the importance of using multi-task learning for improving the utilization of the learned features.

Finally, we propose a novel approach for explaining the performance of smaller but efficient network architectures (*e.g.* EfficientNet[12]) based on the compress-ability of the encoded feature layer. We thereby show that smaller networks are able to perform on-par with bigger networks (with higher parameter count) by increasing the utilization of the joint encoded features.

Future Work. This paper raises an interesting question which might provide interesting directions for future research in this area. First, we note that the proposed analysis in Section 5.2 assumes a negative correlation between network compressibility (from distinctiveness pruning) and equivalent feature utilization. That is, we argue that the number of non-distinct feature nodes (within threshold (θ_{thresh})) is inversely correlated with the feature utilization of the given layer. However, we also note

that some architectures (*e.g.*VGG-11) purposefully encourage less distinctive representations by using the dropout layers. Dropout[10] randomly drops a given feature node during the training process, which implies that using dropout encourages learning representations where any feature node when dropped can be compensated by the remaining features. This goes against our notion of encouraging distinctiveness to maximize feature utilization. This coupled with the results from Section 5.2 suggests the need for discarding the dropout layers while designing high performance network architectures.

References

1. Brock, A., De, S., Smith, S.L., Simonyan, K.: High-performance large-scale image recognition without normalization. arXiv preprint arXiv:2102.06171 (2021)
2. Gao, Y., Bai, H., Jie, Z., Ma, J., Jia, K., Liu, W.: Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11543–11552 (2020)
3. Gedeon, T.D., Harris, D.: Progressive image compression. In: [Proceedings 1992] IJCNN International Joint Conference on Neural Networks. vol. 4, pp. 403–407 vol.4 (1992). https://doi.org/10.1109/IJCNN.1992.227311
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
5. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25**, 1097–1105 (2012)
6. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European conference on computer vision (ECCV). pp. 116–131 (2018)
7. Pasunuru, R., Bansal, M.: Continual and multi-task architecture search. arXiv preprint arXiv:1906.05226 (2019)
8. Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J.: Efficient neural architecture search via parameters sharing. In: International Conference on Machine Learning. pp. 4095–4104. PMLR (2018)
9. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
10. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15**(1), 1929–1958 (2014)
11. Standley, T., Zamir, A., Chen, D., Guibas, L., Malik, J., Savarese, S.: Which tasks should be learned together in multi-task learning? In: International Conference on Machine Learning. pp. 9120–9132. PMLR (2020)
12. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning. pp. 6105–6114. PMLR (2019)
13. Tan, M., Le, Q.V.: Efficientnetv2: Smaller models and faster training. arXiv preprint arXiv:2104.00298 (2021)
14. Yao, Y., Zheng, L., Yang, X., Naphade, M., Gedeon, T.: Simulating content consistent vehicle datasets with attribute descent. In: ECCV (2020)
15. Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., Finn, C.: Gradient surgery for multi-task learning. arXiv preprint arXiv:2001.06782 (2020)
16. Zhang, Y., Yang, Q.: A survey on multi-task learning. IEEE Transactions on Knowledge and Data Engineering (2021)