CS 182 Final Project: Computer Vision

Lily Bhattacharjee, Varun Dashora, Esha Madhekar

1 Introduction

1.1 Background

The CS 182 Computer Vision (CV) Project is an image classification task based on the ImageNet dataset, involving identifying objects in images from a subset of the data – TinyImageNet. Spanning 200 classes of 500 64x64 RGB training images each, the aim of the project is to accurately classify normal images that resemble the training set and the 10,000 validation images provided, while also optimizing performance on perturbed images.

Despite being relatively low resolution (64x64), accurate results on TinyImageNet are significant because new model architectures may be generalized to real-world computer vision challenges. Because the images and the dataset themselves are relatively small and thus require less time to train, but at the same time internally complex and more difficult to overfit than MNIST or CIFAR-10, applications of deep neural networks in this context can range from self-driving cars to smart warehouses, automating the human eye's process of deriving multidimensional data from a scene. Implementing a model with high validation and test accuracy robust to naturally-occurring perturbations in object orientation, size, etc. can offer insight into overfitting-resistant layer orders and operations.

1.2 Model Comparison Metrics

To evaluate models relatively by effectiveness, we observed the following factors:

- Number of epochs to attain a benchmark training accuracy ($\approx 30\%$), rate of increase over epochs
- Peak validation accuracy (before overfitting)
- Execution time per epoch, prioritizing efficient models
- Performance on model-specific adversarial dataset

1.3 Summary of Results

Our optimal model was a mode ensemble of 20 GhostNets, each trained on the entire training set, and a subset of an adversarial set generated by Fast Gradient Sign Method. This resulted in the highest overall validation (64.39%) and adversarial (77.63%) image accuracy. Our hypothesis, which involved replacing portions of existing ResNet-50 and Dual Path Network architectures with Ghost modules, a plug-and-play cell meant to introduce redundancy in convolutions without losing significant image information (see section 3.3 for more background), didn't perform as well as expected, and these trained models were not included in the final submission.

2 Literature Survey/Related Work

Previous work in developing architecture tailored to analyzing 2D images, specifically in ImageNet competition submissions, has included ResNets, Dual Path Networks (DPN), and GhostNets. Deep networks commonly encounter the vanishing gradient problem (VGP): some gradient elements become exponentially small, slowing or halting parameter updates and the network's learning. This occurs because partial derivatives in the backpropagation algorithm are calculated with the chain rule, making it more likely that small weights shrink further as the number of layers increases (He et al., 2015). Early contributions to the ImageNet challenge, including AlexNet and VGG – while revolutionary at the time – are no longer considered state-of-the-art because the VGP limits model depth and thereby information capacity (He et al., 2015).

ResNets, which introduced skip connections, allow networks to learn residual functions based on the layer inputs, rather than the sequential series of linearities and nonlinearities of simple feed-forward network (He et al., 2015). DPNs build on this advance by proposing a hybrid design that combines the benefits of ResNets and DenseNets to reuse and explore new features, allowing good representations to be learned more quickly on shallower DPNs (Chen et al., 2017).

GhostNets were designed out of a necessity for efficient architectures with competitive accuracies on ImageNet (Han et al., 2020). ResNet-50, a common baseline model for the dataset, encompasses $\approx$ 25.6M parameters, and requires a large amount of computing power to train until convergence, and to output results for a batch of inputs (Han et al., 2020). Because some CV applications, such as self-driving, require fast real-time feedback, suggested improvements such as pruning unimportant weights, weight and activation quantization, and knowledge distillation led to performance upper-bounded by the unmodified network's accuracy (Han et al., 2020). GhostNets mitigate this issue by generating more features per layer, using fewer necessary parameters, which decreases computational complexity by introducing map redundancy (Han et al., 2020).

3 Background

3.1 ResNets

The main concept underlying ResNet success is the architecture-agnostic residual connection. Assuming that $H(x)$ is the learned representation of a few layers in a ResNet, the layers should equivalently be able to learn the residual function $H(x) - x$, a linear combination of the inputs and the previous nonlinear transformation (He et al., 2015). Instead of allowing sequences of layers to learn $H(x)$, ResNet layers learn $F(x) = H(x) - x$. Introducing a skip connection between the first and last layers of the block such that the input is propagated to the end allows the network to recover the same output $H(x) = H(x) - x + x$, although learning in the new structure is easier to optimize (He et al., 2015).

Deep networks without skip connections encounter a degradation problem, in which increasing depth initially leads to increasing accuracy that saturates and then decreases, because later layers might be best represented by identity functions that are difficult to learn by fitting a stack of layers rather than overfitting (He et al., 2015). Optimally, in a residual model, it is easier to learn a residual of 0, allowing the input to the first block layer to take a shortcut to the last (He et al., 2015).

3.2 Dual Path Networks

The DPN improves on the residual path (element-wise addition of input features to the output of a block) developed by ResNet and the densely-connected path (input and output feature concatenation) suggested by the DenseNet architecture, which in-

dependently produced state-of-the-art ImageNet accuracy (Chen et al., 2017). DPNs are designed to exploit the benefits of both – feature reuse and exploration – while requiring fewer parameters and computational resources (Chen et al., 2017). DPNs are composed of micro-blocks: 1x1 conv, 3x3 conv, 1x1 conv (Chen et al., 2017). The output of the block is linked to two paths, with one part element-wise added to the residual connection while the other part is concatenated with the input feature maps (Chen et al., 2017). Because the overall architecture of the DPN is very similar to a ResNet, the densely-connected paths can be added on as slice and concatenate layers without measurable computational or memory cost.

3.3 GhostNets

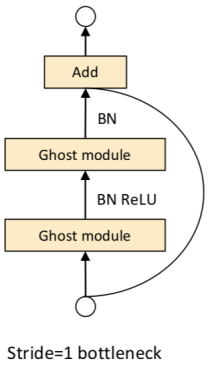

Figure 1: Ghost bottleneck architecture

The basic unit of a GhostNet is the Ghost module, which takes advantage of the existing feature map redundancy in CNN layers to reduce convolutional filter resource usage (Han et al., 2020). A convolutional layer for input $X \in \mathbb{R}^{c\times h\times w}$ can be written as $Y = X * f + b$ where f is the filter $\in \mathbb{R}^{c\times k\times k\times n}$ (Han et al., 2020). As the filters and channel sizes increase in deeper networks, the number of floating point operations needed explodes (Han et al., 2020). Because of pixel overlap in adjacent convolutional operations, output feature maps can be very similar, and represented as a combination of some intrinsic feature maps and their "ghosts" (Han et al., 2020).

In a Ghost module, m (controlled by hyperparameter s) feature maps can be created using a smaller convolution $Y' = X * f'$, $f \in \mathbb{R}^{c\times k\times k\times m}$, which reduces the number of FLOPs to $m \times h' \times w' \times c \times k \times k$ (h' and w' are output dimensions), a quantity that grows much more slowly than n (Han et al., 2020). The remaining $n - m$ feature maps to obtain an output with the correct dimensions, can be generated with cheap linear operations on the m intrinsic maps (Han et al., 2020).

3.4 Project Improvements

The GhostNet, composed of serial Ghost modules, is hypothesized to have generalized improved results. Although the GhostNet architecture suggested is only based on MobileNet's structure, the general idea of replacing convolutional blocks with Ghost modules of the same size should theoretically produce comparable accuracies with cheaper computations. By exploiting convolutional layer redundancy and introducing the Ghost module's cheap operations to approximate the similarity without explicitly calculating all convolution output values, we hypothesized that the modified networks would take fewer epochs to train and converge, without a significant decrease in accuracy. In this project, we explore the performance of ResNet-50 and DPN-68 after integrating Ghost modules into the following layers:

ResNet-50

- all 3 bottlenecks in layer 1's feed-forward sequence
- all (3 + 4 + 6 + 3 = 16) bottlenecks in all 5 layers

DPN-68

- all 4 stride-1 convolutions in dual path block 1
- all 49 stride-1 convolutions in all 5 dual path blocks

3.5 Adversarial Data Augmentation

Other than preliminary data transforms, including randomly jitter of colors and hues, axial flips, rotations, arbitrary grayscale, and affine transformations (e.g. shears), we considered improving model robustness to adversarial inputs by generating auxiliary datasets with Generative Adversarial Networks (GANs). A GAN can be used to create more images of a specific class or mix of classes, and is composed of a generator and a discriminator.

In our use case, the discriminator takes image-integer pairs as inputs, with each image classified as fake or real – from the input dataset – while the generator attempts to create images that minimize loss over the course of a zero-sum game. After training the GAN for 90 epochs, the produced images were noisy and indistinguishable, likely due to the GAN instability with low-resolution images, and thus weren't used in training the final models. Implementing the fast gradient sign method

Figure 2: Green frog output

Figure 3: Red frog output

Figure 4: Visible frog eyes in output

(FGSM) generated a usable set of adversarial images. FGSM, as a white box attack, uses the target network's gradients to create an image that maximizes loss while introducing perturbations small enough to fool a trained model with static parameters, as shown in Figure 5. After training the final ensemble models on clean images, the model states were used to create 3,200 images each, and new models of the same architecture were re-trained with mini-batches of 5 adversarial inputs per iteration.

4 Methods and Results

4.1 Baseline: ResNet-50

Our baseline model was a ResNet-50 pretrained on ImageNet. The only modification made was altering the number of classes in the final layer to 200 instead of ImageNet's full 1,000. Starting with the pretrained weights, we trained the model further on TinyImageNet images. The reason the network was slow to train, only reaching 30% accuracy after 1 epoch, is likely because the original weights were fine-tuned to ImageNet's 224x224-sized images, which have higher resolution. The training accuracy appears to logarithmically increase and taper over

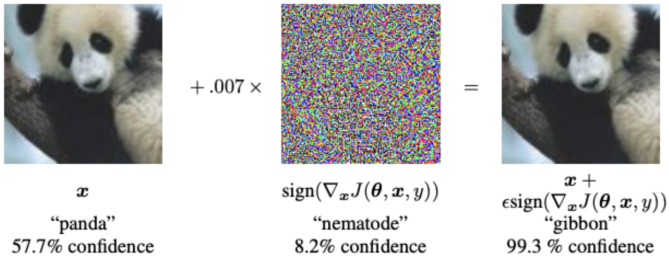

Figure 5: FGSM procedure example on panda

Figure 6: In contrast to the GAN outputs, FGSM outputs visually resemble clean inputs, even though none of the images would be classified as "frog" by the target model (DPN, GhostNet, EfficientNet, ResNext-WSL respectively).

as the number of batches increases. ResNet accuracy has empirically been observed to rise fastest during the first few epochs; even though the model parameters can take many epochs to converge completely, the accuracy does not change significantly (He et al., 2015). As seen in Figure 8, beyond the first epoch,

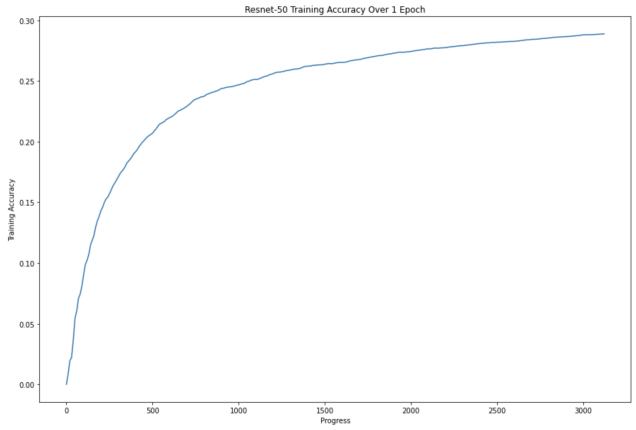

Figure 7: ResNet-50 Training Accuracy Over 1 Epoch

ResNet-50's training accuracy doesn't increase monotonically, instead spiking up briefly in the first few batches and decreasing steadily from there. This behavior can be explained by the initial batches in the first epoch contributing the most to setting the weights, while later batches have a smaller effect due to the combined impact of all of the preceding batches. The resulting model best fits images with features similar to the first batches, which indicates the utility of ensemble models, in which the best-fitting batches will be selected randomly, allowing for better coverage of the dataset than a single model.

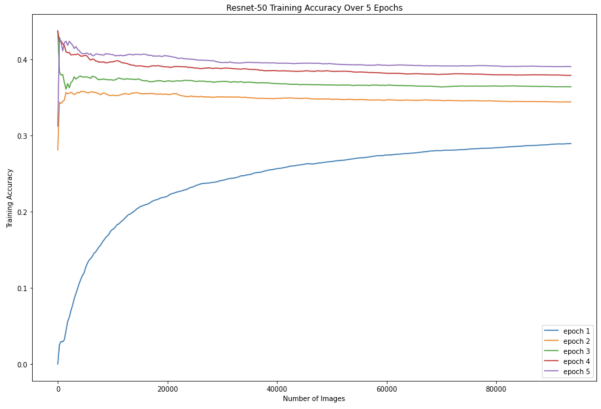

Figure 8: ResNet-50 Training Accuracy, 5 Epochs

4.2 Model Evolution

The next models trained included (1) a ResNet-50 in which the first layer of the network included bottlenecks that were replaced with Ghost bottlenecks (Ghost-1-ResNet-50), (2) a ResNet-50 with all bottlenecks replaced with Ghost bottlenecks (Ghost-All-ResNet-50), (3) a pretrained DPN-68, and (4) a DPN-68 with all convolutional blocks replaced by Ghost bottlenecks (Ghost-All-DPN-68), as described in Section 3.4. From Figure 10, Ghost-1-ResNet-50, Ghost-All-ResNet-50, and Ghost-All-DPN-68 performed very similarly to each other, reaching 4-5% training accuracy over 1 epoch and continuing to linearly increase to an asymptote of $\approx 25 - 30\%$.

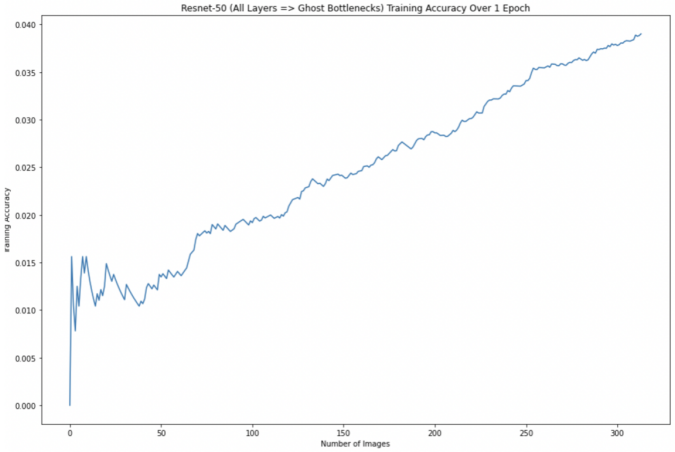

Figure 9: Ghost-All-ResNet-50 Training Accuracy, 1 Epoch

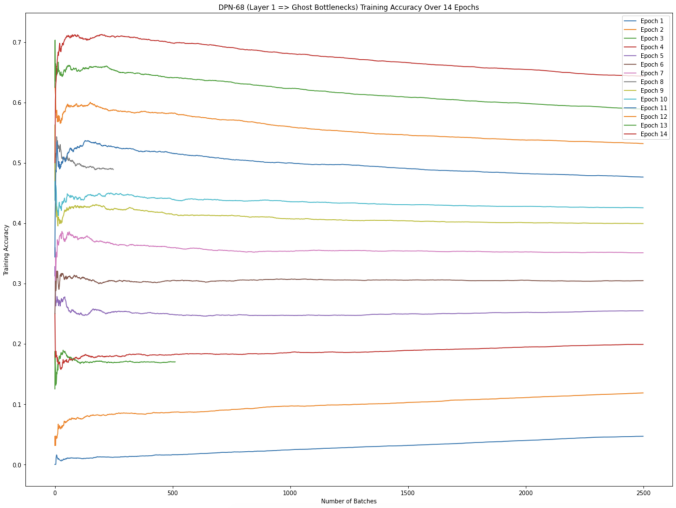

Figure 10: Ghost-All-DPN-68 Training Accuracy, 1 Epoch

As Figure 11 shows, after around 16-20 epochs, like Ghost-All-DPN-68, all 4 models began to overfit, as training accuracy continued to increase slowly, reaching 60-70%, while validation accuracy began to fluctuate at around 40%. This was far below the reported state-of-the-art performance of even the regular pretrained models, which was surprising. Because we initially weren't aware that the validation images were indistinguishable distribution-wise from the training set, we attributed lower than expected accuracy to either lack of robustness to adversarial images, hypothesizing the weight optimization had fallen into a local minimum that an adjustable learning rate would help by converging to a global minimum more quickly, or the low resolution of TinyImageNet images, which would make it more difficult to extract features. For the last reason, we assumed for an extended period that higher accuracy wouldn't be possible given the lower content of information in the data. Additionally, after realizing that transforms weren't being applied evenly between training and validation sets, and fixing this bumped Ghost-All-DPN-68's peak accuracy up to 50.04%. Noticing that

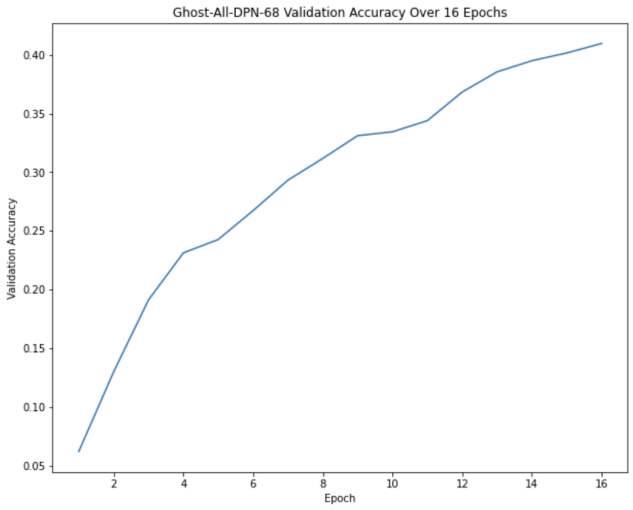

Figure 11: Ghost-All-DPN-68 Training Accuracy Over 16 Epochs

Model	Training
DPN-68	52.1%
ResNext-WSL	35.17%
Ghost-All-DPN-68	50.04%
DenseNet	6.91%
GhostNet	24.11%
ResNet-50	35.6%
GhostNet-Free	64.74%

Table 1: Ensemble 1 Individual Model Performances

Model	Training	Validation
DPN-68	64.24%	52.19%
GhostNet	78.99%	65.52%
EfficientNet	79.50%	70.06%
ResNext-WSL	57.98%	62.37%

Table 2: Ensemble 2 Individual Model Performances

models were performing similarly well ($\approx 50 - 60\%$ accuracy), we created ensemble models to mitigate each individual model's blind spots and reach a consensus output for each image. We experimented with mode, mean, and weighted mean-style (in which all models were trained together, with weights evolving according to how often their predictions were correct on training data, similar to AdaBoost) ensembles, and the former was most successful. Expectedly, mean and weighted mean ensembles don't make much sense to implement on categorical data e.g. votes for class 68, 66, and 64 should not be averaged to 66, which explained the resulting low validation accuracies (4.258% for weighted mean). A weighted mode average ended up reducing to the contributions of a few models only for all inputs, and resulting in 60.90% validation accuracy, not significantly different from the mode ensemble with an accuracy of 56.08%.

Ensemble 1 represents a model that calculates the mode of the outputs for each component model. Ensemble 2 performs the same consensus operation while considering a set of high-accuracy models and performing an additional resizing operation to transform 64x64 images to 224x224 and interpolating in bicubic mode. We started with partially trained component models for architectures that were slow to converge. The performance of Ensemble 1 is worse than the highest performing model (GhostNet-Free) as a result of the high variance in validation accuracies, but because Ensemble 2's models are closer in training and validation accuracy, Ensemble 2 outperforms EfficientNet. We noticed models in a mode ensemble needed to glean different features from the images to be diverse enough to improve validation when grouped together (fulfilled by varying architectures), but also well-trained so as not to introduce noise or inaccurate predictions into the consensus. Note that the difference between GhostNet and GhostNet-Free is that all parameters were free to compute gradients in the latter. Ensemble 1's and 2's validation accuracies were 56.08% and 72.75% respectively.

4.3 Final Model

Regular GhostNet and EfficientNet without modifications took the fastest time to train, and – as theoretically expected – reached higher accuracies within fewer epochs. Honing in on GhostNet, our final model involved training baseline DPN-68, EfficientNet, GhostNet, and ResNext-WSL for 2-3 epochs to reach 60%+ training accuracy, and then using FGSM to generate 3,200 targeted adversarial images per model.

Our submission is an ensemble of 20 GhostNets, where each GhostNet i is a mode 'expert' on the entire training dataset and adversarial images corresponding to classes $10 * (i - 1)$ to $10i$. Individual GhostNet component models were much faster to train than the alternatives, taking around 30 min / epoch, and experts could be trained in parallel. Each GhostNet obtained $\approx 59 - 60\%$ training and $\approx 55\%$ validation accuracy, and the final ensemble reached 64.39% validation and 77.63% adversarial image accuracy.

5 Conclusion

After exploring the possibility of inserting GhostNet modules into selected bottlenecks and convolutions in existing state-of-the-art models like ResNet-50 and DPN-68, it was surprising to discover that these cells do not contribute significantly to information capacity and performance seems upper-bounded by the original architecture. It is likely that Ghost modules worked well with MobileNetV3 because this architecture already includes expansion and reduction blocks similar to the structure of the first variant of the Ghost bottleneck. Additionally, because Ghost modules lose image features by introducing cheap operations, they're likely to work best on large, high-resolution images. Even after resizing the images to 224x224 ImageNet size with interpolation, lower accuracy can be attributed to increased information loss as deeper layers are replaced with Ghost layers.

Our final submission is based on a discovery about the robustness of ensemble models: to balance the tradeoff between clean and adversarial image accuracy, it may be a good idea to train each component model on a set of combined majority clean and minority noisy images. This makes each model very good at classifying unperturbed validation images because the adversarial images are not a large part of the training set, while the other models which are each trained on a small segment of adversarial images compensate for that, each becoming an 'expert' on a certain set of classes. Taking the mode of the model outputs mitigates overfitting and model variance, allowing the ensemble to perform better than its best model.

6 Team Contributions

Lily Bhattacharjee

- trained the following models and generated visualizations for accuracy over epochs: Dual Path Network, ResNext-WSL, EfficientNet, GhostNet, ResNet-50, VGG-16, Ensembles 1 & 2
- explored the possibility of using ghost cells to increase model information capacity
- wrote almost all of the classification-related code to process training and validation images to create torch datasets, execute the training loop, save intermediate model states after each epoch, support ensemble models composed of different architectures
- built on Varun's adversarial image generation code to create a set of 3,200 perturbed images per model
- trained the final ensemble (mode of 20 GhostNets) model submission
- set up the folder structure, code and README for the final submission, including modifications to test_submission.py

Varun Dashora

- implemented the FGSM algorithm and generated 1,536 (no class labels) and 200 (with class labels) adversarial inputs, 1 for each TinyImageNet class
- explored other methods of adversarial image generation, including GANs and corruptions from ImageNet-C
- significant work on GANs – attempted at least 20 different architectures

Esha Madhekar

- significant work on modifying InceptionNet-V3 dimensions and architecture
- trained the following models: InceptionNet-V3 (not included over model evolution due to image sizing issues), DenseNet, ResNext-WSL
- explored training ResNext-WSL with ghost cells, DenseNet & GoogleLeNet combination (similar to InceptionNet)

Effort distribution: 45% (Lily)-30% (Varun)-25% (Esha)

7 Acknowledgments

We would like to acknowledge our use of the following pre-trained models, which were retrained on TinyImageNet data:

- ResNet-50
- DPN-68
- GhostNet
- EfficientNet
- VGG-16
- DenseNet-161
- ResNext-WSL
- InceptionNetV3

Also integral to the success of this project was GhostNet code and model architecture from the following link (repository: ghostnet.pytorch, author: d-li14).

8 Bibliography

Hendrycks, D., Mu, N., Cubuk, E., Zoph, B., Gilmer, J., & Lakshminarayanan, B. (2020). AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty. Proceedings of the International Conference on Learning Representations (ICLR).

Ian J. Goodfellow, Jonathon Shlens, & Christian Szegedy. (2015). Explaining and Harnessing Adversarial Examples.

Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, & Chang Xu. (2020). GhostNet: More Features from Cheap Operations.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. (2015). Deep Residual Learning for Image Recognition.

Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, & Jiashi Feng. (2017). Dual Path Networks.