Computer Vision
Deep Learning Report

XUE YU (Zoe YUU), Trinity College Dublin, Computer Science – Augmented and Virtual Reality

1. Introduction and motivation

Image classification problem is “one of the core problems in Computer Vision”, which has “a large variety of practical applications” and serves as the fundament of many other Computer Vision tasks, such as object detection, segmentation etc. [1].
Since AlexNet achieved 15.4% top 5 error rate on ImageNet ILSVRC challenge in 2012, many breakthroughs on accuracy have been made. In 2015, Residual Net (ResNet) with 152 layers has achieved 3% error rate, which is better than the result made by the human judges [1].
However, despite the great performance convolutional neural networks displayed, they are also known to be difficult to train due to the complexity introduced by the large amount hyperparameters, and problems caused by the exponentially vanishing gradients. [1].

Because of the importance of image classification problem and its complexity, in this study, I’m motivated to first design a new model named “Extractor Team (E.T.)” based on the architecture and basic block of ResNet and Inception net. Then, to implement this model, to train it on Tiny ImageNet – a subset of famous ImageNet, containing 200 classes[1]. Furthermore, observations on E.T.’s performance are to be made in experiments of adjusting hyperparameters and structures. Finally, the top 1 and top 5 accuracy of E.T. will be compared to the performances of the state-of-art models - VGG16 and ResNet50, both pre-trained and retrained from scratch. Additionally, attempts will be made to understand E.T. based on the confusion matrix generated based on the predictions made by the model.

2. Theoretical background

To make deeper neural network easier to be optimized, inventors of ResNet proposed to address the degradation problem by “feedforward neural networks with shortcut connections”. [2] (Fig.1)

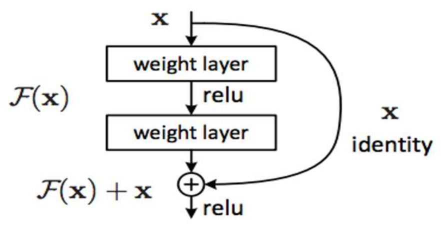

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig.1 Residual module, a building block

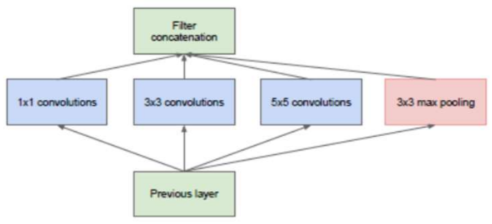


Fig.2 Inception module, naive version

To find the optimal local structure, the inventors of Inception net suggested an architecture with convolutional layers with 1x1, 3x3, 5x5 filters, and then concatenate

[1] 500 training images, 50 validation images and 50 test images per class

the outputs into a single output tensor, which “forms the input of the next stage”. [7] (Fig.2)

Inspired by the philosophy of these two state-of-art models, and taken account of the limited time and computation resources, the plain version of the model proposed in this paper, codenamed “Extractor Team (E.T.)[1] has the following architecture:
The basic model consists two varieties:
Block x: built by 3 convolutional layers using 1x1x16 filters, output has the shortcut added in to resemble the design of ResNet.
Block y: built by 3 convolutional layers using 3x3x16 filters[2], output has the shortcut added in[3].
One module consists one block x and one block y, stacking upon each other; the entire model is constructed with 4 identical modules, followed by a global average pooling layer, one fully connected layer with “ReLU” as activation function and one 200-way fully connected layer with “softmax”. The total number of weighted layer is 27. (Fig.3)

3. Implementation

E.T. has been implemented using Keras and Tensorflow with CUDA8 and cuDCNN6, chosen for Keras’ reputation as widely applicated API in both research and production fields [2]. The experiments have been conducted on a computer armed with NVIDIA GeforceGT1080 on Windows 10.

4. Experiments, analysis and result discussion

Since training a plain version of E.T. for one iteration on a 200 classes dataset takes 481-530s[4], experiments have been firstly conducted on subset of Tiny ImageNet due

[1] The name derives from the film “Inception”, in which “extractors” plant and extract information from people’s brain.

[2] The number of filters -16 - is the “base” number suggested when wide ResNet has been created, in which a network’s width in the first block is designed as 16xk and determined by factor k. [8]

[3] In the original design, input has been processed by 3 branches:
one branch has 3 convolutional layers using 1x1 filters, another has 3 convolutional layers using 3x3 filters, again another is the shortcut connection proving identity mapping.
It is hoped that the first branch can extract local information limited in a small range, the second branch can extract features which can only be detected by larger patches, and the third can provide the unextracted, original information. All those branches should be combined by concatenating their outputs into one output vector, followed by a convolutional layer with 1x1 filters to learn to adjust the weights of 3 outputs from 3 branches.
However, experiments show that such a structure takes too much time to train - ca. 3 minutes more per epoch, the architecture has been then adjusted to use two branches per block only and using block x and y alternatively.
Also, in the adjusted architecture, originally a batch normalization layer and a scaling layer has been added to every convolutional layer to provide a regularization effect. They were disabled later during the experiments because, like it has been already claimed in the paper and been observed during the experiment, this architecture requires heavy data augmentation to work, which in turn requires much more time to converge and to generate results. [8]

[4] The time needed for one epoch varies even when the implementation and the data fit in remain unchanged, which is a phenomenon remaining unexplained.

to the time and computation limitations. If the results showed clear improvement of the performance of the model, the same strategy would be then applied to the model trained on the full 200 classes dataset to make further test.

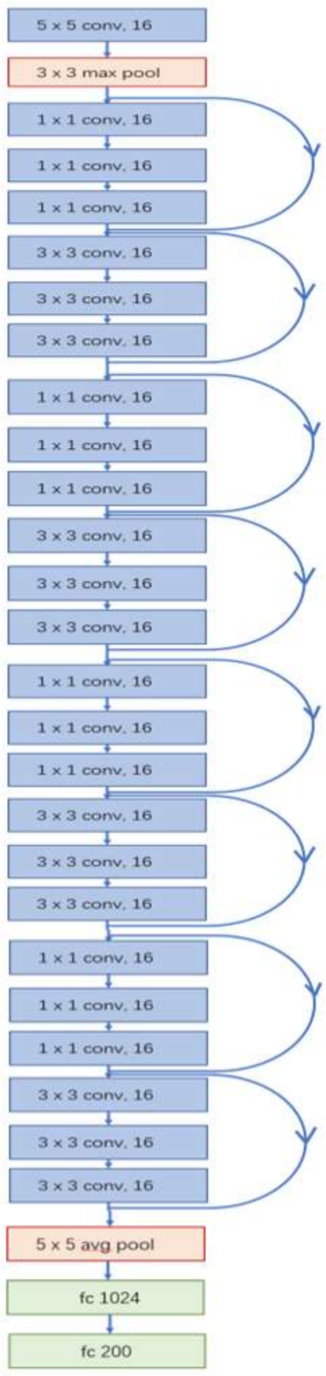


Fig.3 Extractor Team network

Throughout the experiments, the evaluation metric chosen has been “accuracy”. Moreover, unless been specially pointed out, the activation function used is “ReLU” and the loss function used is the “categorical cross-entropy” provided by Keras. Furthermore, batch size of the input training data has also been adjusted during the experiments with the sole goal to minimize the training time for one iteration, its impact on the converging time hasn’t been considered.

Firstly, two state-of-art models have been trained as baselines. Then, E.T. has been tested on a small subset of Tiny ImageNet containing 2 classes, after successfully overfitting this small dataset, following experiments have then been carried out.

4.1 Experiment on optimizer and learning rate schedule

Because different optimizer has shown the different impact on VGG16 and ResNet50 during the training for the state-of-art models, the first experiment has been conducted on a subset of 10 classes, with the aim to find out the optimal optimizer for E.T. and the proper learning rate schedule.

Curiously, the optimizer RMSprop seems cannot train E.T. properly at all: the initial learning rate has been scanned over the range from 0.1 to 1e-6, the training accuracy has been raised to 0.49, whereas the validation accuracy stays at 0.05 - the accuracy of a random guess and haven't been improved at all since training begins[1].

The optimizer has been then switched to SGD, the initial learning rate has been scanned over from 0.1 to 1e-6, momentum has been set to 0.9, decay e-6. The result shows that E.T. begins to be slowly trained. After 800 epochs, the training accuracy stabilized at 0.18 whereas the validation accuracy stabilized at 0.22.

The experiment went on with Adam as the optimizer. The initial learning rate has been scanned over the range of 0.1 to 1e-6. The training schedule has been set that if the training loss hasn't been improved in 50 epochs, it will be reduced to half of the current learning rate. Results show that, on a subset of 10 classes, with the initial learning rate of 1e-3, E.T.'s performance has been improved to validation accuracy of 0.55 before flatten lined.

In all the experiments afterwards, Adam has been used as the optimizer for E.T., the initial learning rate has been set to 1e-3 with a schedule to reduce learning rate to its half if the training loss hasn't been reduced in 50 epochs.

4.2 Experiments on dropout layer and image augmentation

Dropout has been "adopted by many successful architectures" to prevent overfitting, in which it "prevents units from co-adapting too much" by randomly disabling a certain portion of the units from the network during training. [6,8] Mostly, Dropout layers are "applied to the top layers", to "prevent feature coadaptation and overfitting".[8]

The experiment of the impact of this classical technique on E.T. has been firstly conducted on a subset of 5 classes.

Firstly, a plain version of E.T. has been quickly trained as the baseline model, which, as expected, began to overfit after 15 epoch and the validation accuracy remains at 0.66.(Fig. 4)

[1] An unpleasant result observed but couldn't be explained yet.

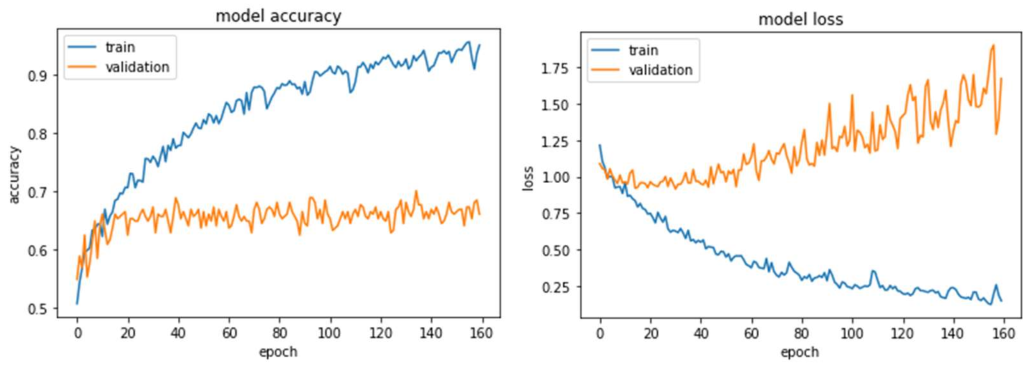


Fig.4 Plain E.T. trained on subset containing 5 classes

After this, an E.T. model with dropout (0.5) layer has been added before the last fully connected layer.(Fig. 5)

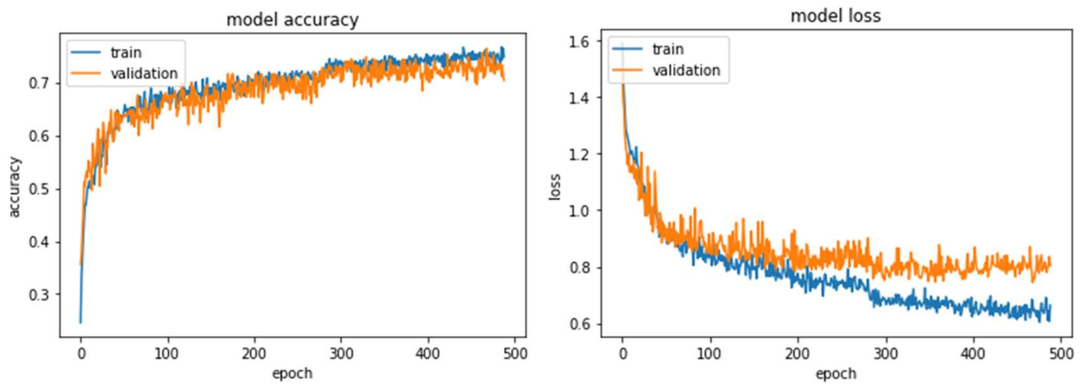


Fig.5 E.T. with dropout (0.5) before the last fully-connected layer, trained on subset containing 5 classes

Although it has taken 400 more epochs to converge, the model with a simple dropout layer has achieved a validation accuracy of 0.72 and loss of 0.8 before overfitting. The result shows that, at least on a small dataset, even a simple dropout layer can prevent overfitting effectively and improve the validation accuracy up to 24%.

Therefore, experiments have been continued, in which the performances of the models with and without dropout layer have been evaluated on 200 classes. The result shows that the plain model without dropout layer begins to show the sign of overfitting when validation accuracy reached 0.22, whereas the model with dropout layer shows no sign of overfitting until validation accuracy reached 0.27-0.28.

Image augmentation is another effective way mentioned to fight against overfitting, the theory is to “fool” the model to make it think that a larger input training set has been provided by randomly transforming a portion of the input images without adding more data in the dataset. [1]
Experiments on image augmentation have been conducted by firstly, applying 8 different augmentations randomly to the input training data: rotation, width shift, height shift, shear, zoom, channel shift, horizontal flip, vertical flip. With these image

augmentations, the model can reach a validation accuracy of 0.70-0.71 and loss of 0.68[1]. (Fig.6)

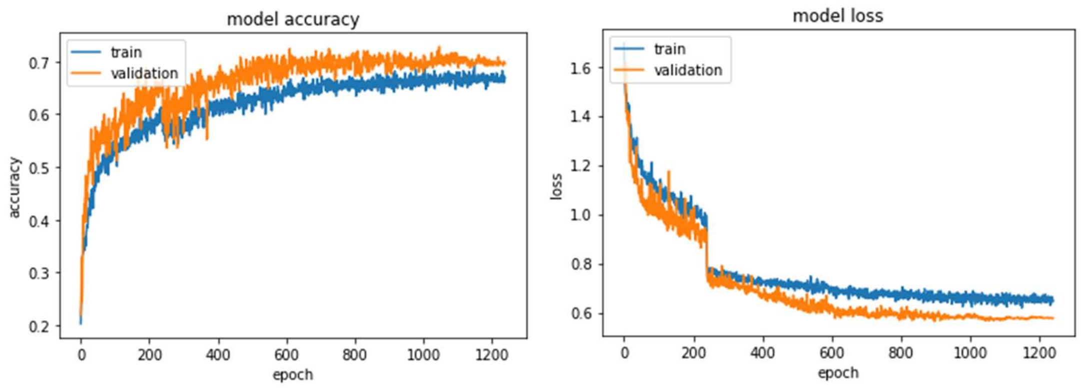


Fig.6 E.T. with 8 image augmentations, trained on subset containing 5 classes

The result shows that besides adding dropout layer, image augmentation is also a powerful tool to prevent an early overfitting. However, unlike the simple dropout method, image augmentation is much more time consuming: on a small subset of 5 classes, it takes 900 epochs to converge (500 epochs more than the model with one dropout layer), also, it takes 6 more seconds per epoch for the model with image augmentation to train (17s comparing to model with dropout layer's 11s), and converged at a slightly worse validation accuracy by 0.02.

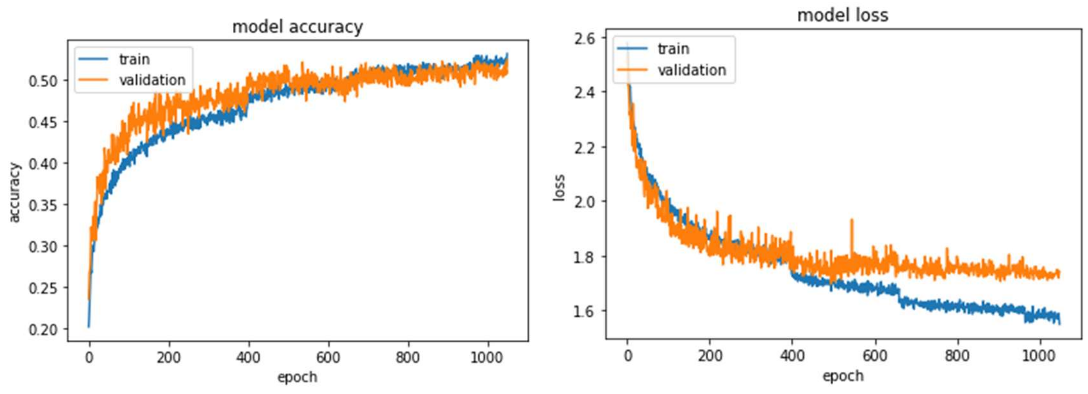


Fig.7 E.T. with dropout layer and 5 image augmentations, trained on subset containing 20 classes

Encouraged by the positive impact dropout layer and image augmentation brought, combined two methods (dropout layer 0.5 before the last fully connected layer, and five different image augmentations: rotate, width shift, height shift, horizontal flip, vertical flip), the model has been trained on a subset of 20 classes. Without much monitoring of the training process, the model has reached a validation accuracy of 0.525. (Fig. 7)

[1] Noticed that there is the inconsistency in both figures. Possibly because the experiments on image augmentation has been interrupted by the training for Kaggle competition. Later, as the experiments resumed, part of the training history (ca. 100 epochs) has been missing, which might explain the abrupt drop of the loss at the 200 epochs. Moreover, as the experiment resumes from the weights saved, learning rate has been reset to the original learning rate of 1e-3, whereas in the previous training, learning rate has been reduced according to the learning rate schedule, which might explain the drop of accuracy at the beginning of epoch 200.

The experiment has been then continued on the full set of 200 classes, with the same dropout layer and five image augmentations used in the previous experiment, the model reached a validation accuracy of ca. 0.29 before hit the plateau[1]. (Fig. 8)

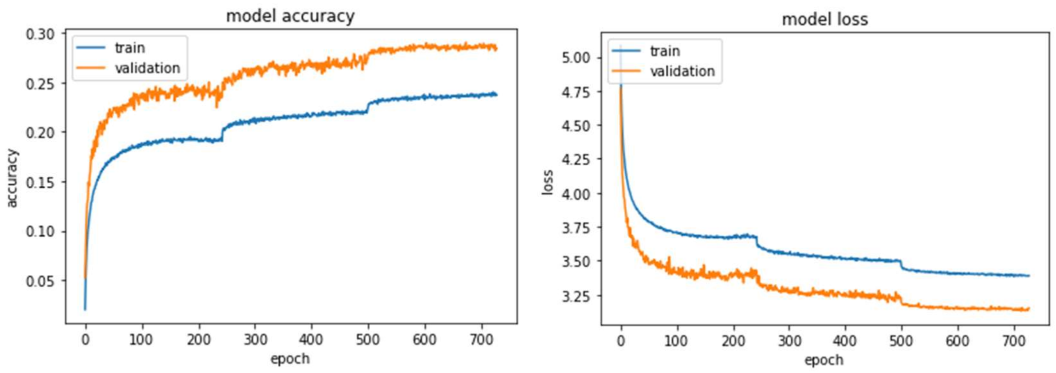


Fig.8 E.T. with dropout layer and 5 image augmentations, trained on subset containing 200 classes

Based on this version of E.T. but stopped using any image augmentation[2], the model later entered Kaggle competition has been trained. It has reached a validation accuracy of ca. 0.34[3] before the improvement of validation accuracy became very slow: for the last 0.05 validation accuracy improved, it took 35 epochs to train. (Fig.9) Although the result shows that the model is still learning, due to the time limitation, this training has not been continued.

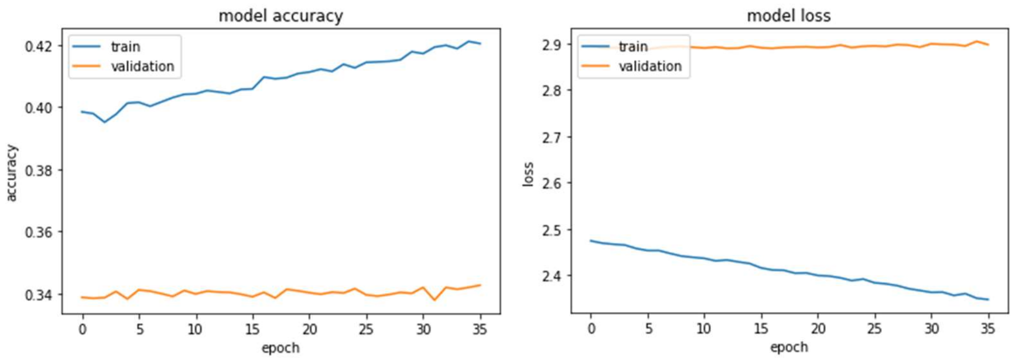


Fig. 9 E.T. with dropout layer, trained on 200 classes (result of the last 35 epochs in total 172 epochs training)

4.3 Experiments on various structures: width, depth and strides

For ResNet, research shows that widen its blocks provides a “much more effective way of improving performance” “compared to increasing their depth”. [8] Since E.T. also uses identity mapping and has a similar structure with ResNet, it is to expect that widen its blocks can also improve its training efficiency.

[1] The abrupt change in curves should be explained by the reduction on learning rate caused by learning rate schedule.

[2] Because image augmentation method takes more time to train (it takes at least 580s to complete training for 1 epoch, comparing to 481s-530s/epoch training time for the plain model with dropout layer) and in order to improve the performance faster to catch the Kaggle competition’s deadline, image augmentation has been dropped in the training this version of E.T.

[3] The best weights saved for this model is top 1 validation accuracy of 0.3470.

Experiments on the width of block has been conducted firstly on a subset of 5 classes, the number of filters has been changed from 16 to 32[1]. At least on a small set, the advantage this strategy brought is clear to observe (Fig.10):

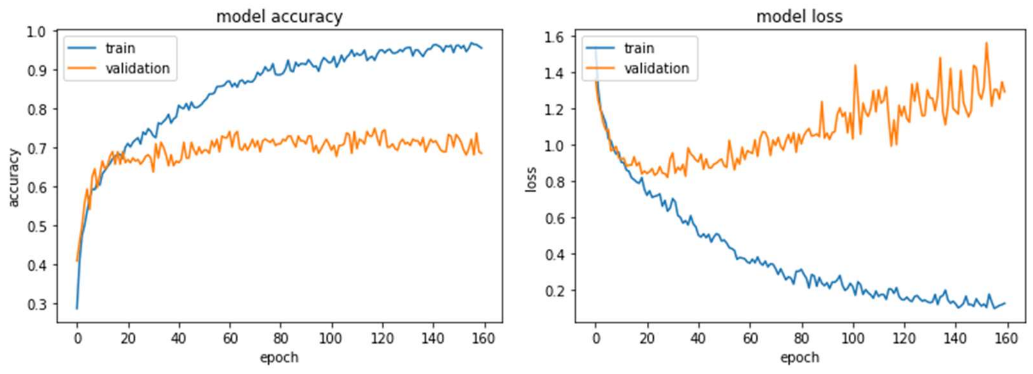


Fig. 10 wide E.T., trained on subset containing 5 classes

The wider model (without dropout layer or any image augmentation) has achieved validation accuracy of 0.7 at ca. 20 epochs, whereas the plain model has only reached a validation accuracy of 0.66. Although the training time for each epoch is 2s longer than the plain model (13s vs 11s), the wider model is still much more effective than the original one.

When the same architecture has been applied on the dataset of 200 classes (with dropout layer and five image augmentations), compared to the original model, in the first 50 epoch, the wide model has archived 0.01 more validation accuracy. However, due to limited time and computation resources, this experiment has not been carried out to the end to see how much validation accuracy the wide model on class 200 can achieve.

Depth has been considered to be "of crucial importance" for CNN models, since many leading results on the ImageNet challenge have been achieved by very deep models. [2,7] Experiments has been conducted on this strategy as well, in which the plain model's depth has been doubled by simply doubling the number of modules. The result shows that, on a subset of 5 classes, without dropout layer or any image augmentation, a deep E.T. model (with 49 convolutional layers compared to a plain E.T. model's 25 convolutional layers) reached a validation accuracy of ca 0.68 at 23 epoch, however, it also needs more time to complete one epoch (22s to plain E.T. 's 11s, and wide E.T. 's 13s). (Fig.11)
In conclusion, although a deeper E.T. is capable to reach a slightly better validation accuracy (2% better than the plain model), at least on a small dataset, it is not as effective as a wider model.

[1] k value has been set to 2. [8]

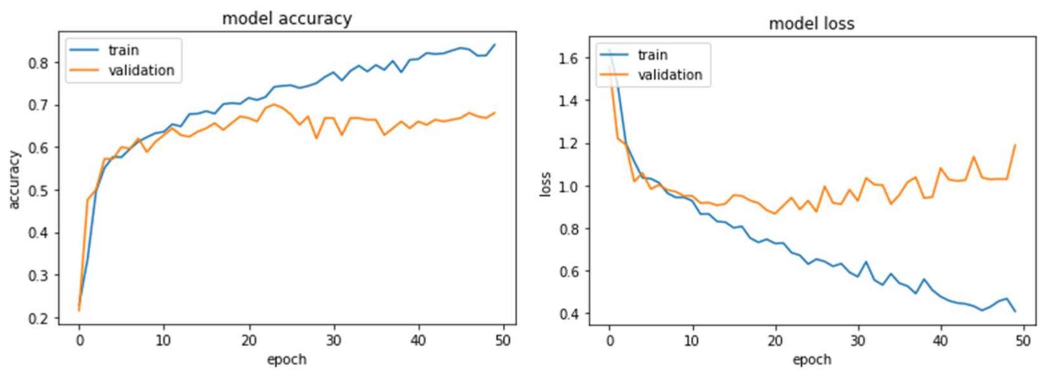


Fig. 11 deep E.T., trained on subset containing 5 classes

The creators of Inception nets used max pooling layer with stride 2 to halve the resolution of the grid. [7] Experiment on stride has also been included, in which the max-pooling layer after the first convolutional layer has been used with stride 2. The plain model has reached only a 0.38 validation accuracy, which is lower than the plain E.T. mode by 0.28. This unpleasant result is, however, not unexpected. The hypothesis is that on images of such small size - 64x64 pixels, using filters on stride 2 – especially at the very beginning, omitted too many details which in turn causes training to be less effective. (Fig.12)

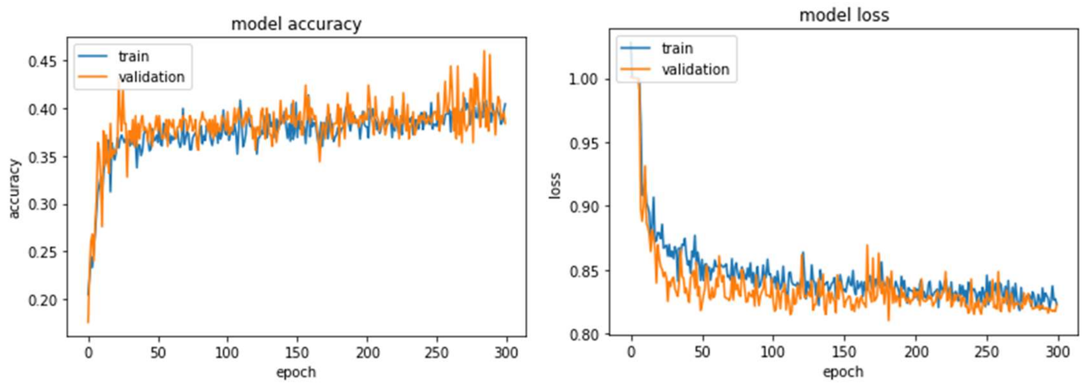


Fig. 12 E.T. with max-pooling layer with strides 2, trained on subset containing 5 classes

4.4 Experiments on evaluation metrics: Loss functions

Since the objective of the training is to minimize the loss, consequently, choosing different loss functions should have impacts on model's performance. Experiment has been made in which the current loss function: "categorical cross-entropy"[1] has been changed to "categorical hinge". Instead of encouraging "the log probability of the correct class to be high", this loss function should encourage "the correct class to have a score higher by a margin than the other class scores", [1,4] The modified model has been used to train on a small subset of 5 classes. (Fig.13)
The result shows that on a subset of 5 classes, the model has converged at a validation accuracy of 0.6, which is 0.06 lower than using the loss "categorical cross-

[1] Since the output represents a probability distribution, using this loss function we measure the distance between what the model believes this distribution should be, and what the ground truth claims it should be [4,5]

entropy”. At least on this smaller subset, cross-entropy loss function helps E.T. plain model achieves a better performance.

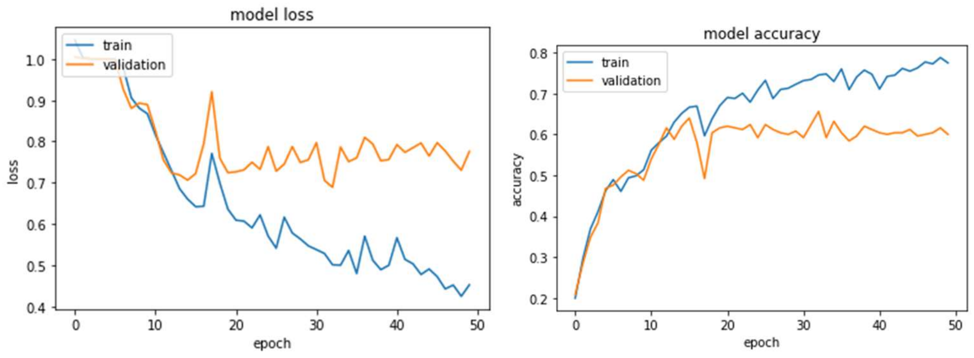


Fig. 13 E.T. with loss function “categorical hinge”, trained on subset containing 5 classes

4.5 Experiment on activation functions

Experiment on the model with different activation function has been conducted, in which the model with activation function “tahn” instead of “ReLU” has been used to train on the subset of 5 classes. (Fig.14)

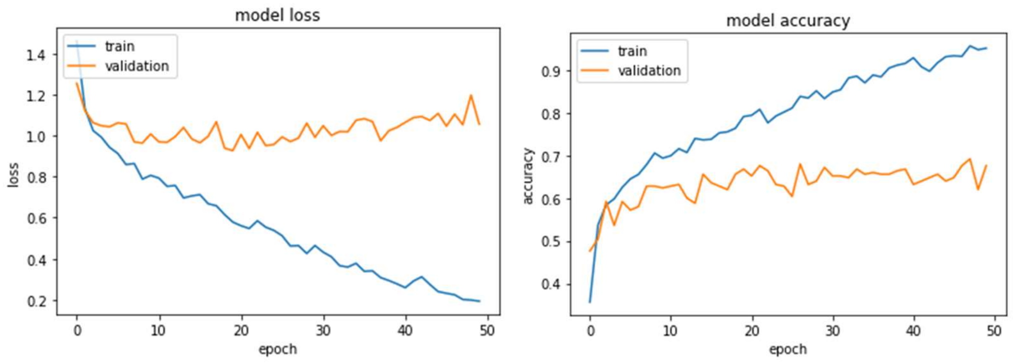


Fig. 14 E.T. with activation function “tahn”, trained on subset containing 5 classes

The result shows that at least on a subset of only 5 classes, activation function “tahn” makes the plain model converge ca. 5 epoch faster than “ReLU”, although it has converged at a slightly lower validation accuracy of 0.65 and the curve of validation accuracy seems more unsteady. It would be interesting to observe such model’s performance on the dataset with 200 classes.

5. Understand E.T. and comparison

5.1 Comparison with state-of-art models

The E.T. model taken part in the Kaggle competition has achieved 0.3470 top 1 validation accuracy and 0.5509 top 5 validation accuracy. However, this model still hasn’t converged[1], whereas during the experiments an E.T. model with dropout layer,

[1] Not to oversee, either, that this model has used the pre-trained weights from E.T. trained with image augmentation first, then been continuously trained without image augmentation. The facts above lead to the conclusion that simply comparing this model’s performance with state-of-art models’ performance would explain nothing at all. However, as pointed out above, it is impossible to newly train an E.T. model with dropout layer and image augmentation again due to limited time and computation resource. Therefore, the results achieved by models trained on 20 classes have also been included below.

trained on a subset containing 20 classes has converged and has achieved a validation accuracy of 0.525.
Therefore, a VGG16 model, pre-trained and retrained with five image augmentations, using RMSprop as optimizer with the initial learning rate of 1e-3, and a ResNet50 model, pre-trained and retrained with five image augmentations, using SGD as optimizer with the initial learning rate of 1e-4, no momentum, decay e-6, on a subset of 20 classes and on 200 classes have been used to compare with the top 1 and top 5 validation accuracy both E.T. has achieved.

	ON 20 CLASSES		ON 200 CLASSES	
	Top 1 Validation Accuracy	Top 5 Validation Accuracy	Top 1 Validation Accuracy	Top 5 Validation Accuracy
E.T. (WITH DROPOUT)	0.525	0.838	0.347	0.551
VGG16 (PRE-TRAINED)	0.456	0.892	0.520	0.794
VGG16 (RETRAINED)	0.531	0.854	0.442	0.782
RESNET50 (PRE-TRAINED)	0.663	0.909	0.623	0.821
RESNET50 (RETRAINED)	0.562	0.873	0.541	0.801

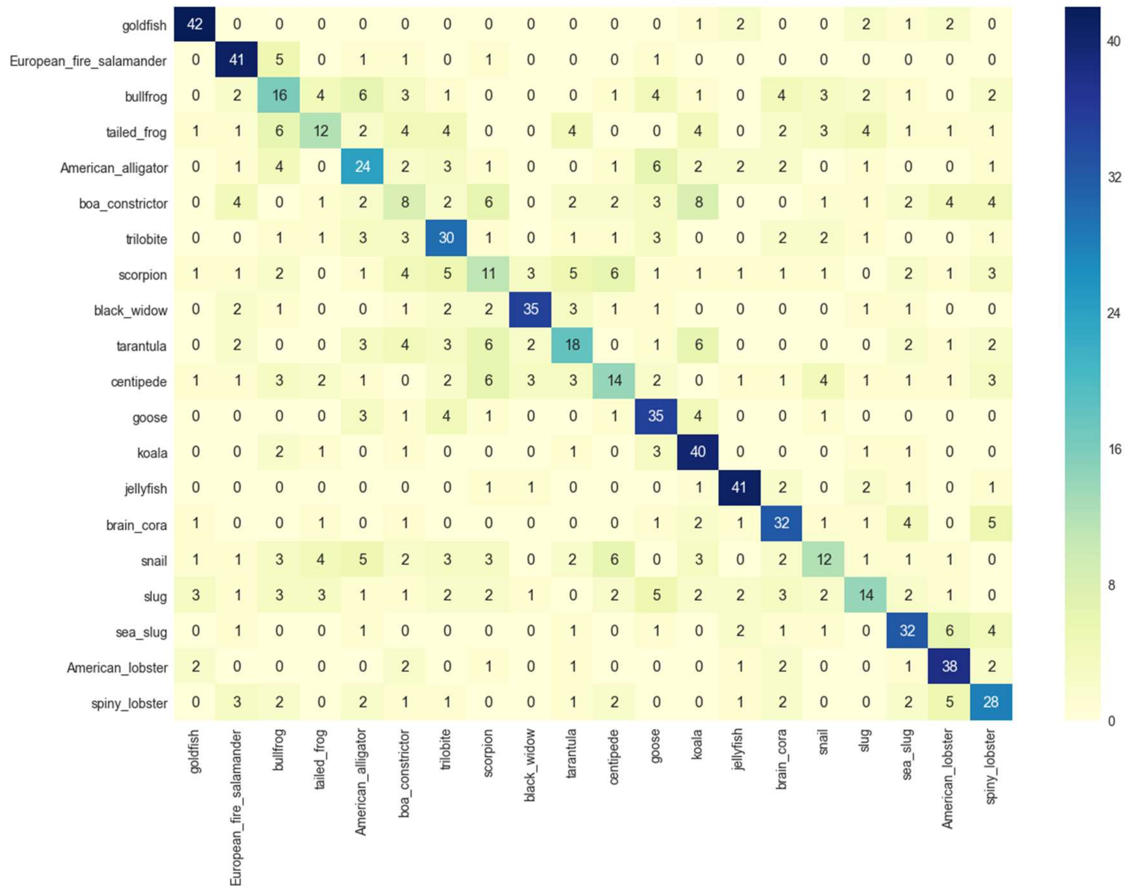
Noticed that since Tiny ImageNet is a subset of ImageNet, on which the state-of-art models provided by Keras have been trained. However, the validation accuracies of the state-of-art models trained here are 0.1-0.2 lower than the accuracy published on Keras' official site.[4] One possible explanation is that each class only has 500 training data, which is not sufficient for a complex convolutional neural network to learn perfectly. [3] To address this problem, much heavier image augmentation than the five augmentations applied here should be used to improve the validation accuracy further. It is also possible that the model has only hit the plateau, given more training epochs it might come out of the plateau and the performance might be able to improve again, this hypothesis can only be verified by continuing training the model for more epochs to observe their performance.[1]

Also notice that although VGG16 has much simpler structure than E.T., retrained VGG16 has achieved 0.006 better top 1 accuracy than E.T. when been trained on the subset, which, in my opinion, is the proof that deeper complicated convolutional neural networks are more difficult to train due to issues such as the degrading problem, or co-adaptions between blocks etc.
On the other hand, although sharing the similar basic structure and being deeper, the retrained ResNet50 has also displayed better performance than E.T., one plausible reason for this is that the official ResNet50, containing 64, 128, 256, 512 filters for each block, is significantly wider than E.T. which only has 16 filters in every block, and being wider has been proved to be an efficient tool to improve accuracy.

[1] Also noticed that due to time limit and inexperience at the beginning of experiments, the learning rate schedule and optimizer of those state-of-art models have not been fine-tuned, which might explain the clearly low top 1 accuracy of VGG16 trained on subset of 20 classes.

5.2 Confusion matrix

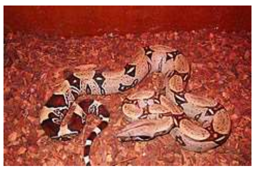
To further understand E.T., a confusion matrix generated from the predictions made by the converged model trained on subset of 20 classes has been plotted. (Fig.15)



	goldfish	European_fire_salamander	bullfrog	tailed_frog	American_alligator	boa_constrictor	trilobite	scorpion	black_widow	tarantula	centipede	goose	koala	jellyfish	brain_cora	snail	slug	sea_slug	American_lobster	spiny_lobster
goldfish	42	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	2	1	2	0
European_fire_salamander	0	41	5	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
bullfrog	0	2	16	4	6	3	1	0	0	0	1	4	1	0	4	3	2	1	0	2
tailed_frog	1	1	6	12	2	4	4	0	0	4	0	0	4	0	2	3	4	1	1	1
American_alligator	0	1	4	0	24	2	3	1	0	0	1	6	2	2	2	0	1	0	0	1
boa_constrictor	0	4	0	1	2	8	2	6	0	2	2	3	8	0	0	1	1	2	4	4
trilobite	0	0	1	1	3	3	30	1	0	1	1	3	0	0	2	2	1	0	0	1
scorpion	1	1	2	0	1	4	5	11	3	5	6	1	1	1	1	1	0	2	1	3
black_widow	0	2	1	0	0	1	2	2	35	3	1	1	0	0	0	0	1	1	0	0
tarantula	0	2	0	0	3	4	3	6	2	18	0	1	6	0	0	0	0	2	1	2
centipede	1	1	3	2	1	0	2	6	3	3	14	2	0	1	1	4	1	1	1	3
goose	0	0	0	0	3	1	4	1	0	0	1	35	4	0	0	1	0	0	0	0
koala	0	0	2	1	0	1	0	0	0	1	0	3	40	0	0	0	1	1	0	0
jellyfish	0	0	0	0	0	0	0	1	1	0	0	0	1	41	2	0	2	1	0	1
brain_cora	1	0	0	1	0	1	0	0	0	0	0	1	2	1	32	1	1	4	0	5
snail	1	1	3	4	5	2	3	3	0	2	6	0	3	0	2	12	1	1	1	0
slug	3	1	3	3	1	1	2	2	1	0	2	5	2	2	3	2	14	2	1	0
sea_slug	0	1	0	0	1	0	0	0	0	1	0	1	0	2	1	1	0	32	6	4
American_lobster	2	0	0	0	0	2	0	1	0	1	0	0	0	1	2	0	0	1	38	2
spiny_lobster	0	3	2	0	2	1	1	0	0	1	2	0	0	1	2	0	0	2	5	28

Fig. 15 Confusion matrix, generated from the predictions made by E.T. with dropout layer and image augmentation, trained on 20 classes

The matrix shows, for instance, E.T. has a general problem to recognize "boa constrictor'', and easily confuses it with "scorpion", or "koala".



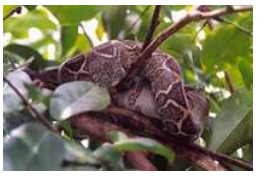


Fig. 16 Correct label "boa constrictor", predicted label "scorpion" Fig.17 Correct label "boa constrictor", predicted label "koala"

It is possible that the context (Fig. 17) or the twisted form of the constrictor has made the prediction more difficult[1] - especially after the input images have been rotated and flipped.

On the other hand, although the result shows that "Sea slug" is relatively easier to be recognized, but it can often be confused with "American lobster", or "spiny lobster". It might because some sea slugs have relatively unique forms (Fig.19) or colors (Fig.18), which has misled the model to make wrong predictions.

[1] However, the confusion matrix plotted when the validation accuracy of 0.29 shows, "koala" seems to be E.T.'s favorite prediction, almost all the classes has been confused with "koala" a lot.

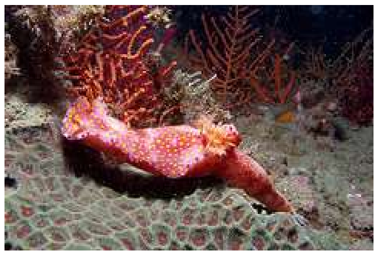


Fig.18 Correct label “Sea Slug”, predicted label “American lobster”

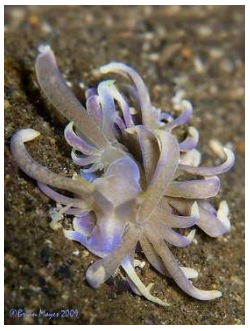


Fig.19 Correct label “Sea Slug”, predicted label “spiny lobster”

6. Summary

Since the breakthrough made by AlexNet in 2012, many powerful models have been invented and fine-tuned to address image classification problem. Although the leading result has mostly been holding by deeper models, a deep convolutional neural network is also known to be difficult to train, therefore, research and experiments have been conducted to address these problems.

In this study, a CNN model codenamed E.T., modified from the plain version of the identity block in ResNet and Inception net has been proposed, experiments on modifying structures and hyperparameters has been conducted in the attempts to improve the training efficiency. The performance of the E.T. trained on both 200 classes and a subset of 20 classes has been evaluated in comparison to the validation accuracy achieved by state-of-art models. Also, attempts have been made to understand E.T. further by analyzing the result shown by confusion matrix.

7. Future work

The experiments couldn’t be included or properly completed due to time and computation source limits are as follows:

- The inventors of the wide residual nets also added a dropout layer into each block between convolutional layers and after ReLU to prevent it from overfitting. [8] The experiment should be conducted in which dropout layer (e.g. dropout (0.1)) to be inserted in the same place in E.T.’s block to observe the impact of this method.
- Much heavier image augmentation should be applied on E.T. when being trained on 200 classes: in the original experiments, 8 image augmentations have been applied to make E.T. trained on subset of 5 classes to reach a validation accuracy of 0.70-0.71– which is 0.5 better than the plain model. Due to limited time and computation resource, the E.T. trained on 200 classes only used 5 image augmentations at first, then no image augmentation at all, it should be expected that heavier augmentation than this could help the model to converge to a higher validation accuracy.
- Since experiments show that widening model is an effective tool to increase training effectiveness, the experiment with a wider E.T. trained on 200 classes should be continued until the model converge. Moreover, other

values of factor k should be applied. [8] For instance, try to use convolutional layers with 64 filers (k=4) in the experiment and to evaluate its performance[1].

- A wider E.T. with heavy image augmentation should be trained on 200 classes. After this version of E.T. converged, the final validation accuracy should be used to be compared with the performance of the state-of-art models.
- Experiments on filter size should be conducted, in which 1x1, 3x3, 5x5 filters to be used in different blocks alternatively.
- E.T. with activation function “tahn” should be used to train on 200 classes to evaluate its performance further, since using on a small subset this method has achieved a decent result, even though the performance seems less steady.

References

[1] CS231N. Convolutional Neural Networks for Visual Recognition. http://cs231n.github.io/

[2] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2016)

[3] KERAS BLOG. Building powerful image classification models using very little data https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html (2016)

[4] KERAS DOCUMENTATION. https://keras.io/

[5] PLUNKETT, K., ELMAN, J., Exercises in Rethinking Innateness. A Handbook for connectionist simulations. The MIT Press (1996)

[6] SRIVASTAVA, N., HINTON, G. E., KRIZHEVSKY, A., SUTSKEVER，I., AND SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. Jouranal of machine learning research 15, 1 (2014) pp. 1929-1958

[7] SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D.， AND VANHOUCKE, V., RABINOVICH, A., Going deeper with convolutions. arXiv preprint arXiv:1409.4842 (2014)

[8] ZAGORUYKO S., KOMODAKIS N., Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)

[1] ResNets also double the number of filters when the input size halves. [2] However, in E.T., due to the small input size of the image (64x64), its size is not designed to be reduced to half during each block, therefore, the filers' number also remains the same in each block.