Convolutional Neural Networks for Galaxy Zoo Challenge of Morphological Class Probablity Regression

Yu Cao
New York University
yc3390@nyu.edu

Lei Chen
New York University
lc3909@nyu.edu

Abstract

This report presents our method with multiple-layer convolution neural network to work on the task of Galaxy Zoo Morphological Classification score regression challenge [1]. The task is a regression problem of 37-answer of probabilities of galaxy shape based on image, which is to minimize the RMSE (Rooted Mean Square Error) with respect to the crowd-sourced probabilities. The methods are based on 2 backbone networks of Resnet and plain CNN respectively. As with previous literature on the task, we focus on the rotation robustness of the network and utilize probability constraints to better regress the score. Different loss function criteria are also tried, among which L2 loss performs best. To research into the rotation invariability of the network, we applied GrouPy convolution [2] and defined a metric of rotational robustness for verification. A coarse-to-fine training methodology is proposed for a promotion of the performance of the backbone net. We show that by utilizing our proposed methods, a single model can have an RMSE of 0.07520 on validation set, which has gone beyond the SOTA 0.07671. By fusing multiple models' results, we have achieved an RMSE of 0.07484, also better than the SOTA result 0.07491 [3].[1]

1. Introduction

Galaxies can be of a great variety of shapes, sizes and colors. The massive collected data of the cellestrial objects are impossible to be manually classified into different morphological inspections. The Galaxy Zoo project aims to build automated classification systems for measuring the morphological parameters of the galaxy via a crowdsourcing strategy on images. Volunteers are asked about a series of questions about the shape of the galaxy in an image.

The Galaxy Zoo chalange is to predict the score of the galaxy's morphological probabilities through images. It is illustrated as Fig.1 shows. An image of a galaxy is queried into multiple questions regarding the shape of it. The answers in total comprises of 37 classes whose probabilities are given by the ratio of the peopele's score. The challenge is held as a competition on kaggle on 2014, with RMSE as the metric of evaluation of the prediction probabilities as equation (1) states. The winner [3] has pushed the RMSE loss on the validation set to be 0.07491.

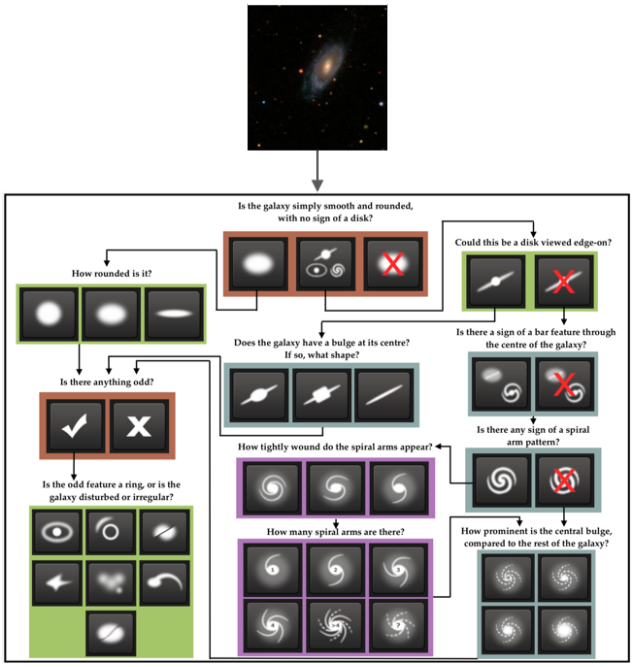


Figure 1: Galaxy Zoo morphological class regression

$$RMSE = \sqrt{\frac{1}{N}\sum_{1}^{N}(p_i - a_i)^2} \qquad (1)$$

[1]Implementation codes can be found at https://github.com/Yucao42/Galaxy_Zoo and https://github.com/leichen2018/Galaxy_Zoo.

In the Galaxy Zoo competition, there are two key challenges regarding the images rotation and the problems' internal normalization constraints of the probability distribution. In this paper, we mainly proposed 2 solutions to the challenge that push the MSE even lower.

1. Adding group equivariant layers to tackle the rotation-invariant problem utilizing filters with a higher dimension of weight sharing.

2. Training models in a coarse-to-fine way so that the model learns the contraints of absolute distribution as well as the conditional distribution of the problem well.

The remained paper is organized as follows. Section 2 gives a detailed description of the key challenges and previous related best solutions on Kaggle. Section 3 proposes the method we have tried. Experimental results are given in Section 4. Finally, the paper concludes in section 5.

2. Key challenges and related work

For general vision recognition tasks, many neural network structures, such as VGGNet [4], ResNet [5], have shown state-of-the-art performances, impressively lowering difficulty of such problems. However, Galaxy Zoo Recognition has two additional challenges, rotational invariance and probability constraints.

Before making our own strategies to tackle these challenges, we firstly review approaches from former Kaggle competition winners' posts, including **Sander** Sedielem, top #1 winner, and Team 6789 (**Tund**), top #3 winner.

2.1. Rotational invariance

Conventional recognition task is sensitive to rotation. For instance, in MNIST dataset [6], '6' can be viewed as '9' with rotation of $180°$, which means the model needs to be rotational invariant. By contrast, in Galaxy Zoo task, impact of rotation should be eliminated because all problems are concerned about galaxy's geometric features instead of its spatial degree as shown in Figure 2.

It is intuitive to re-think about convolutional neural networks. CNN has been proven not having a property of rotational invariance [2]. Actually, combined with pooling layers, CNN pays more attention in transitional invariance since pooling filter is able to extract features from a region of pixels and make deeper layers grasp larger parts' information of input. Since CNN structures are sensitive to rotation, previous winners utilized many methods of data augmentation to tackle the challenge. Sander rotated every training input by $45°$, flipped it and cropped it, so each input turned into 16 images at last. Tund took more options of rotational degrees to random choose. In addition, both of them applied model averaging when evaluating test images. Sander blended 17 models and Tund used 4 models. They achieved great performance promotion after merging single models.

Figure 2: Galaxy Zoo morphological classification scores should be sensitive to rotation.

In practise, most methods of data augmentation and model averaging are of great help to improve test performance. But Sander's method of generating 16 training inputs worked not well for us since it would severely slow down training and evaluating.

2.2. Probability constraints

This network's output for each image is a unconstrained vector coming from a Fully-Connected layer. It represents the 37-class probabilities of all the possible answers shown in Figure 1, which illustrates the decision tree of the morphological questions clearly. Each answer is a node in the decision tree and all cousin node's probabilities should sum up to all their parent nodes' probabilities sum. So for each question, the equation (2) holds in the representation of absolute probabilities. This constraint limits the distribution of the output globally. This helps the model to learn a "coarse" distribution of the task.

$$\sum_i P_{parents} = \sum_j P_{children}. \tag{2}$$

To further learn the distribution, we have to satisfy the local constraints in the form of conditional probabilities. For each question, the conditional probabilities of the answers sums up to 1. Namely, the k-th answer of a question with l answers has the conditional probability P'_k as follows. This "fine" distribution needs a prerequisite of coarse understanding of the problem.

$$P'_k = \frac{P_k}{\sum_l P_l} \tag{3}$$

The SOTA solution by Dieleman [3] has done training applying the absolute probability constraint. However, we propose a coarse-to-fine training that further learns the conditional probability constraint that push the limit even further. Models with previous "coarse" understanding are further fine-tuned by jointly minimizing the MSE of absolute probability as well as conditional probability. That introduces our coarse-to-fine learning schedule which contributes a lot to the MSE loss reduction.

3. Proposed Methods

We use Resnet18 [5] and plain CNN structure by Tund's implementation[2]. The Resnet doesn't give a free lunch of boosting of performance while the coarse-to-fine trianing scheme helped a lot for improving it. We also re-implemented Tund model with extra batch normalization layers after each convolutional layer in pytorch as a baseline of our experiment. Modifications are made on top of the Tund model to make a groupy network.

3.1. Group equivariant convolutional layer

GrouPy layer [2] was introduced as a new design of convolutional layer where higher dimension of weight sharing based on self-defined group transformation is applied to filters. We mainly took a GrouPy layer of the group $p4m$ parameterized as,

$$g(m,r,u,v) = \begin{bmatrix} (-1)^m \cos(\frac{r\pi}{2})) & -(-1)^m \sin(\frac{r\pi}{2})) & u \\ \sin(\frac{r\pi}{2})) & \cos(\frac{r\pi}{2})) & v \\ 0 & 0 & 1 \end{bmatrix}.$$

This group consists of all compositions of translations, mirror reflections and rotation by 90 degrees about any center of rotation in the grid. A $p4m$ function has 8 planar patches, each one associated with a mirroring m and rotation r as shown in Figure. More details are in the original publication [2].

The model structure is shown in Table 1.

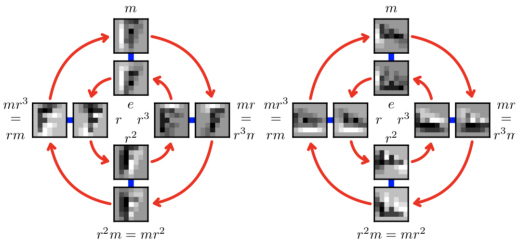


Figure 3: A p4m feature map and its rotation by r.

[2] https://github.com/tund/kaggle-galaxy-zoo.

Table 1: GrouPy model.

Layers	Type	Kernels
0	P4MConvZ2	(3, 8, 5, 5)
1	MaxPooling2d	(3, 3)
2	ReLU(BatchNorm2d)	48
3	P4MConvP4M	(48, 96, 5, 5)
4	MaxPooling2d	(2, 2)
5	ReLU(BatchNorm2d)	96
6	P4MConvP4M	(96, 192, 3, 3)
7	ReLU(BatchNorm2d)	192
8	P4MConvP4M	(192, 192, 3, 3)
9	ReLU(BatchNorm2d)	192
10	P4MConvP4M	(192, 384, 3, 3)
11	ReLU(BatchNorm2d)	384
12	P4MConvP4M	(384, 384, 3, 3)
13	MaxPooling2d	(3, 3)
14	ReLU(BatchNorm2d)	384
15	FC	(8192, 512)
16	ReLU(Dropout2d)	0.5
17	FC	(512, 37)
18	ReLU	37
19	Cust.	(37, 37)

3.2. Customized Layer

The customized layer here is specified as a layer to impose the probability constraint to the output of the neural networks including Resnet-based and Tund-based ones. The $norm$ function force the output obey the global constraint (2) while the $local$ function nomalizes the output locally by constraint (3). Uniformly, the loss is in MSE loss form it performs best among other distance metrics such as KL-divergence [7]. Therefore, the "coarse" loss and the "fine" loss metric are defined as follows. Models are firstly trained with "coarse" loss to for general knowledge on global distribution and are further finetuned through "fine" loss for fine-grained regression on global and local distribution. Here α is a weight factor that adds more attention on local MSE loss. In our training, we typically set α as 3.

$$Loss_{coarse} = \sqrt{\frac{1}{N}\sum_{1}^{N}(p_i - norm(a_i))^2} \quad (4)$$

$$Loss_{fine} = Loss_{coarse} + \alpha\sqrt{\frac{1}{N}\sum_{1}^{N}(p_i - local(a_i))^2} \quad (5)$$

3.3. Training details

3.3.1 Data augmentation

The data augmentation has been an extremely useful method to expand the expressiveness of the data and let the Deep CNN learn better about the distribution of the features in the high-dimensional space instead of overfitting data with noises.

Data augmentation is done in affine transformation and color jittering. Each training image gone through the following data augmentation steps.

Rotation Each image is randomly rotated by a degree within range 360°. And then it is randomly flipped horizontally by a chance of 0.5.

Translation Each image is randomly translated by [-40, 40] pixels in both x and y directions.

Scaling Each image is random rescaled with a scale factor randomly from 1:1.5 and 1.5:1.

Color jittering Each image is augmented in brightness, contrast, saturation and hue within modifying ratio of (0.5, 0.1, 0.1, 0.1) respectively.

Much of the training augmentation detail is consistent with the Dieleman's method [3], while we don't crop and concatenate the images to form a 16 viewpoint extraction as they are too expensive in training.

3.3.2 GrouPy

For GrouPy training, learning rate is set as 0.01 and will be divided by 10 at 100, 150 and 200 epochs when total training has 250 epochs. Weight decay is set to be 0.0005 for regulation term. The model uses SGD as optimization method and is trained from **scratch**. The batch size is set to be 32.

3.3.3 Resnet

For resnet18 training , learning rate is set as 0.02 and will be divided by 10 every 25 steps. Weight decay is set to be 0.0005 for regulation term. ImageNet [8] pretrained model is used to initialize parameters. The batch size is set to be 64 as larger batch seems to result in worse performance. This counters the intuition that big sampling batch will be more close to the real distribution of the data thus leads to a better performance in model [9]. Resnet50 is abandoned as it is slow to train with no performance promotion.

4. Results and observations

This section firstly reports final results of Tund, GrouPy and Resnet. With Tund as the baseline, GrouPy and Resnet both achieved a great promotion on test performance. After adding a customized layer proposed in Section 3.2, single model of Resnet has overgone the SOTA single model as shown in Table5. Model averaging also helped us score better than **stat-of-the-art** for this task in Table 6.

Then, to evaluate rotational invariance of models, we defined a value $R(\cdot)$ to show models' robustness to rotation. It turned out that GrouPy has the best robustness as anticipated and Resnet has worse robustness. This explained why Resnet improved a lot after rotating test images.

4.1. Results of Tund and GrouPy

As a baseline of our experiment, Tund was implemented according to the original post and achieved a test score of 0.08786 that would be top #18 on the Kaggle leaderboard.

Compared with Tund, we implemented a GrouPy model without customized layer at first and approached a test accuracy of 0.08202, which brought a great promotion from the base model. And this score would be top #9 on the leaderboard.

In addition, the GrouPy model with Customized layer achieved test accuracy of 0.07855 from single model and 0.07746 from model averaging, beyond top #2 of the leaderboard. The RMSE training loss curve is shown in Figure 4.

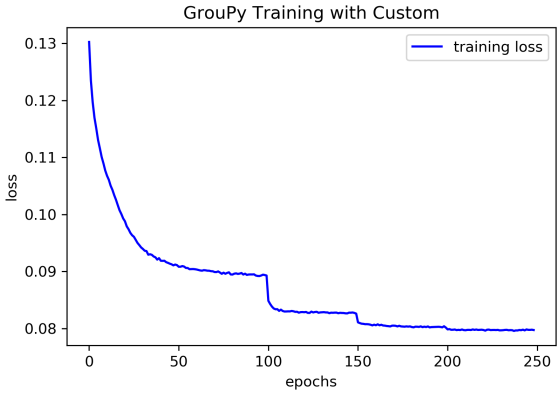
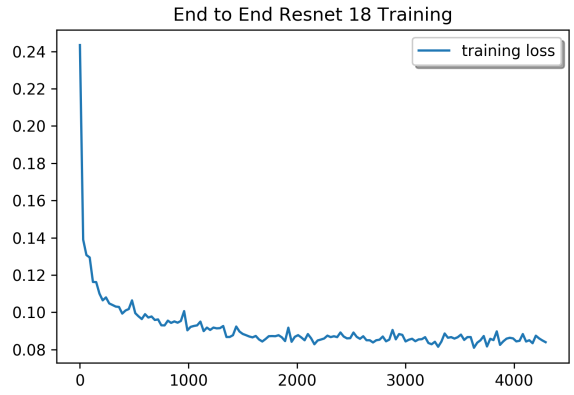


Figure 4: GrouPy training RMSE loss curve

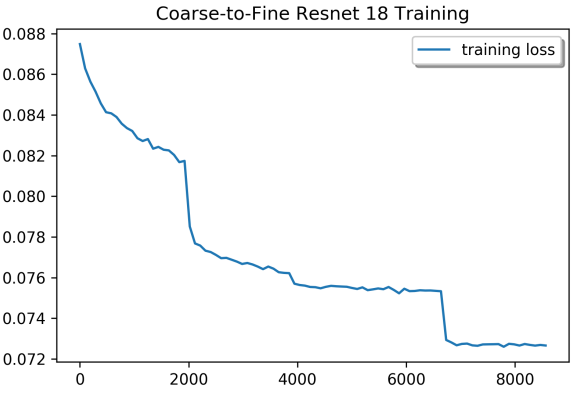
4.2. Results of Resnet

The training curves of RMSE loss in end-to-end fashion and our proposed coarse-to-fine way are presented below. The former "coarse" training converges at 0.0808 and achieved 0.0832 on the validation set, which is far from single model's best performance 0.7671 on validation. Our proposed scheme has boosted the training performance at 0.0726 and is tested on validation of RMSE 0.07521.

From the loss curve we can clearly see that the model learns better by gradually learning global and local distributions. Also experiments are conducted use the "fine" loss directly to train in end-to-end fashion. This results in training RMSE of 0.0798 and validation 0.0812, which only slicely out-performs the "coarse" learning results. So we come to



(a) End-to-end training.



(b) Coarse-to-fine training

Figure 5: Resnet18 training RMSE loss curve

a conclusion that the coarse-to-fine learning schedule contributes to a better results.

Besides coarse-to-fine(C2F) method, we also tried following different schemes of training. None of the following schemes result in a better result so far.

Focal loss Inspired by focal loss in object detection [10], We tried to weight each answer's loss with its proportion on the loss to focus on hard examples. This makes almost no difference with the regular training. The MSE loss barely changes after the focal loss being applied.

Sigmoid A sigmoid layer is put on the end of the network to add non-linearity to the model to generate the score.

Mutual Inspired by mutual learning [11], we try to let resnet-based model and Tund-based model learn from each other by adding a MSE loss between the scores of them.

In general, the training results of resnet-based models are listed on table 2.

Table 2: MSE Loss of Resnet18 Models

CNN module	Training RMSE	Test RMSE
resnet50 baseline	0.0775	0.0807
resnet18 baseline	0.0808	0.0832
resnet18 C2F	0.0726	0.0752
resnet18 Focal	0.0730	0.0769
resnet18 sigmoid	0.0786	0.0809
resnet18 KL	0.9891	0.1350
resnet18 + Groupy Mut	-	0.0825

4.3. Metric of rotational invariance

A metric of rotational invariance is defined to explicitly evaluate a model's practical robustness to rotation. With each test image i as an input, we evaluated 37 probabilities after rotating $0°$, $90°$, $180°$ and $270°$. Therefore, for the i-th image, there are four output vectors V_{ij}, with $j = 1, 2, 3, 4$. Then, with V_{i1} as benchmark, we defined three values $D_i, k = 1, 2, 3$ as follows.

$$D_i = \sqrt{\frac{1}{3 \times 37} \sum_{j=2,3,4} (V_{ij} - V_{i1})^\top (V_{ij} - V_{i1})}. \quad (6)$$

In addition, with each model as a mapping from input image to output vector, a metric $R(model)$ of the model's robustness to rotation is defined as the mean of D_i over all test images, with m as number of images:

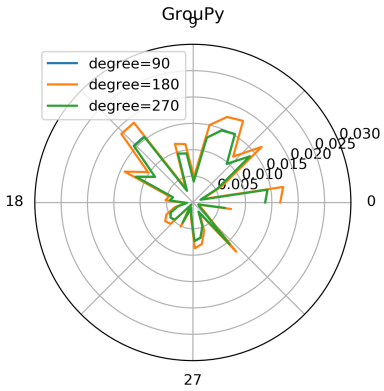
$$R(\cdot) = \frac{1}{m} \sum_{i}^{m} D_i(\cdot). \quad (7)$$

This criterion certainly credits better robustness to rotation with a lower $R(\cdot)$.

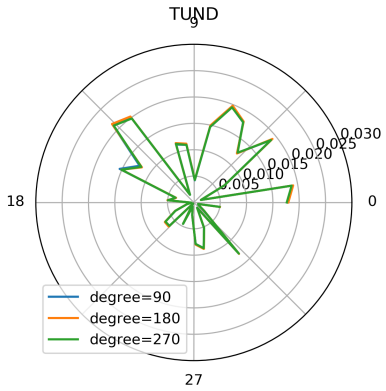
Moreover, a visualization method is introduced to illustrate rotational invariance of all images in each model. Sampled mean of four vectors with four rotation degrees are drawn in one **polar** coordinate system as shown in Figure 6a, 6b, 6c, 6d. The difference between Resent_continuous and Resnet_discrete is data augmentation where the former one used continuous random degrees and the latter one used discrete degrees. So we can expect that latter will have a more uniform results than the former as it was fed with data rotated by such angles more likely.

It is shown in these 4 pictures that **GrouPy** has **better** robustness than all the other models. And two **Resnet** are both more sensitive to rotation than GrouPy and Tund. More accurately, we calculated $R(\cdot)$ of each model as shown in Table 3, which implies the same feature.

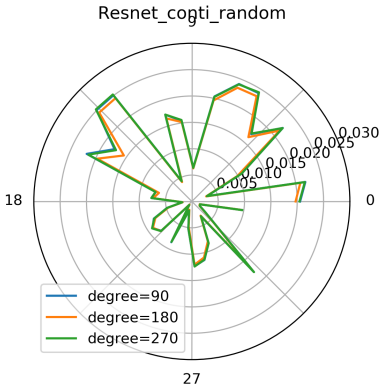
In general, we introduce a simple and intuitive metric to evaluate the rotation robustness of several trained neural networks. And we find that Groupy framework performs best with respect to rotation robustness.



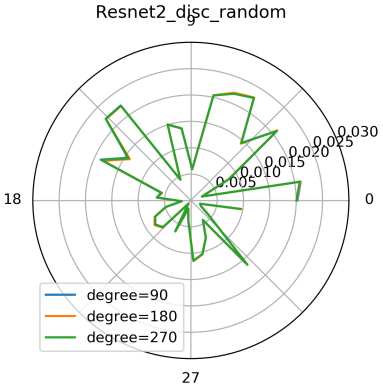
(a) Rotational invariance of all images in **GrouPy** model.



(b) Rotational invariance of all images in **Tund** model.



(c) Rotational invariance of all images in **Resnet_continuous** model.



(d) Rotational invariance of all images in **Resnet_discrete** model.

Figure 6: Rotation invariance of all models.

Model	R(·)
GrouPy	**0.0137**
Tund	0.0152
Resnet_continuous	0.0209
Resnet_discrete	0.0194

Table 3: $R(\cdot)$ of all models. Note that **GrouPy** has best robustness to rotation.

$D_i(\cdot) - D_i(GrouPy)$	Tund	Res18_con	Res18_dis
≥ 0	59.9%	82.6%	77.6%
≥ 0.001	52.7%	78.4%	72.2%
≥ 0.002	45.5%	73.2%	66.2%
≥ 0.004	32.4%	61.7%	53.8%

Table 4: Distribution of gap between rotational invariance between models. Note that both Resnet have weaker robustness than GrouPy and Tund.

5. Conclusions

In this paper of taking the Galaxy Zoo Challenge, we proposed 2 frameworks in tackling rotation invariance and learning probability distributions respectively. Modified groupy networks are trained to show good rotation robustness. In addition, we visualized a metric of rotational invariance and verified GrouPy's advantage on related robustness. Secondly, a coarse-to-fine scheme is proposed and impelemented by a customized layer for better learning the global and local distribution of probabilities. Combining the above-mentioned methods, we push the score beyond state-of-the-art in both single-model and fused results as stated in 5 and 6.

The main challenge of this task is to tackle problems of rotational invariance and probability constraints. GrouPy and customized layers are added and models perform much

Model	Test RMSE
Sander **(SOTA)**	**0.07671**
Top 10 bench	0.08327
Res18 w/o cust.	0.08320
Res18 w/ cust. C2F	**0.07520**
GrouPy w/ cust.	0.07746
GrouPy w/o cust.	0.08202
Tund w/o cust.	0.08786

Table 5: Single model scores

Model	Private Test RMSE
Sander **(SOTA)**	**0.07491**
Team 6789	0.07869
Top 10 bench	0.08303
Resnet + GrouPy	**0.07484**

Table 6: Multi model scores

better on test set, with averaged model going beyond SOTA performance. Moreover, we visualized a metric of rotational invariance and verified GrouPy's advantage on related robustness.

To better improve the performance, we further combine Resnet and up-to-date development of GrouPy on the dataset. However, the training does take more time so we can not reach a conclusion by this paper.

,

References

[1] K. W. Willett, C. J. Lintott, S. P. Bamford, K. L. Masters, B. D. Simmons, K. R. Casteels, E. M. Edmondson, L. F. Fortson, S. Kaviraj, W. C. Keel, *et al.*, "Galaxy zoo 2: detailed morphological classifications for 304 122 galaxies from the sloan digital sky survey," *Monthly Notices of the Royal Astronomical Society*, vol. 435, no. 4, pp. 2835–2860, 2013. 1

[2] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *International conference on machine learning*, pp. 2990–2999, 2016. 1, 2, 3

[3] S. Dieleman, K. W. Willett, and J. Dambre, "Rotation-invariant convolutional neural networks for galaxy morphology prediction," *Monthly notices of the royal astronomical society*, vol. 450, no. 2, pp. 1441–1459, 2015. 1, 3, 4

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 2

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 2, 3

[6] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010. 2

[7] C. Lin, T. K. Marks, M. Pajovic, S. Watanabe, and C. kuan Tung, "Model parameter learning using kullbackleibler divergence," *Physica A: Statistical Mechanics and its Applications*, vol. 491, pp. 549 – 559, 2018. 3

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. 4

[9] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, "Megdet: A large mini-batch object detector," 2017. 4

[10] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 5

[11] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," 2017. 5