2021 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE) | 978-1-6654-2709-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICBASE53849.2021.00067

Animal face classification based on deep learning

Shiwen Han[1,†]
[1]School of Information Science and Technology,
ShanghaiTech University,
Shanghai, 201210, China,
hanshw@shanghaitech.edu.cn

Qiuxin Gao[2,†]
[2]School of Economics, Xiamen University,
Xiamen, Fujian, 361005, China,
1522018220403@stu.xmu.edu.cn

Chaoyu Wang[3]
[3]Faculty of science, Ryerson University,
Toronto, M5B 2K3, Canada,
chaoyu.wang@ryerson.ca

Jiawei Zou[4]
[4]College of automation;
Nanjing University of Posts and Telecommunications
210023 China,
zoujiawei7@gmail.com

[†]These authors contributed equally.

Abstract. Using deep learning to automatically identify and classify wildlife can greatly improve the efficiency of wildlife protection strategies. Based on the public data set of Animal Faces-HQ, this essay analyzed methods to classify different types of animals. Then, this essay developed the four means to predict animal species. First two adopts traditional Convolutional neural networks, while later two construct the deep-learning model based on Auto Encoder. Experimental results showed that our four methods is accuracy and effectiveness while the model in which SVM is superimposed on neural network feature extractor performs slightly better. To verify the applicability of our model, this research applied this model in additional test set of data, and it also achieved satisfactory fitting results.

Keywords: Residual learning; VGG; Auto-encoder; VAE

I. Introduction

Image Clustering and Classifying are the fundamental challenge in machine learning and artificial intelligence. In past few decades, many classical methods such as Resnet and VGG have been proposed to improve the accuracy of image classification. Additionally, deep neural networks can extract the features efficiently and yet can directly added to the traditional machine learning methods, to deal with much complex inputs. Alternatively, deep generative models have shown their power in robust feature representation for challenge unsupervised tasks. So in this project, the project are trying to compare four different deep learning methods between supervised classifying in classical CNN and a variational inference method for unsupervised clustering on a high-resolution image data set.

II. Related Methods

A. Resnet

ResNet[1], which was proposed in 2015 by researchers at Microsoft Research introduced a new architecture called Residual Network. This structure enables the convolutional neural network can be increased considerable depth to improve the accuracy and it solve the problem of gradient disappeared. The model adds layers are identity mapping and the other layers are copied from the learned shallower model, which should have training error no greater than its shallower counterpart. Besides, it is easier to optimize the residual mapping than to optimize the original block which do not have residual parts.

This work adopt cross entropy as the Loss function to updates the parameters in Neural network. The cross entropy is $H(p,q) = -\sum_{i=1}^{n} p(x_i)\ln q(x_i)$. Here $p(x)$ and $q(x)$ are two probability distribution of random variable x. And $H(p,q)$is the cross-entropy of $p(x)$ to $q(x)$.

B. VGG16

In this part of the project, this part focus on VGG16 method. VGG16 uses several 3x3 convolutional layers to replace 5x5 and 7x7 convolutional layers. This can be seen as imposing a regularisation on the 7 × 7 convolutional filters, forcing them to have a decomposition through the 3 × 3 filters (with non-linearity injected in between) [2]. The work will extract the penultimate fully connected layer for feature extraction and input the extracted features into the softmax fully connected layer and SVM classifier, and use this to compare the effects of the two. Support vector machine is an widely used alternative to softmax for classification, and using SVMs (especially linear) in combination with convolutional nets have been proposed in the past as part of a multistage process[3].

C. Unsupervised Clustering with VAE

Unsupervised clustering can be reformed as a dimensional reduction problem which aims to find latent structure of datasets in the low dimension space. Good representations will lead to better clustering results. The deep generative models, including GAN, VAE and FLOW, have attracted much attention for the wonderful abilities to capture the data distribution by neural networks. In another point of view, those methods can be seen as variational inference(VI)[4] , which is one of the key problems in modern statistics. Traditional approximate inference, like MCMC, faces serious problem when handling huge datasets or complex distribution models. VI provide a general mechanism to handle such problems which allow us to explore their applications on more fancy datasets.

This project focus on VAE based methods. From the view of autoencoder, VAE use the encoder and decoder networks to represent posterior and prior and it constrains the latent space with some regularization during the training process. Gaussian

Mixture Variational Autoencoder(GMVAE) [5]use simple linear networks to parametrize the mixture of Gaussians within one training process, which replace the single Gaussian prior of VAE for better representation abilities to fit clustering tasks by nature. Adversarial Auto-Encoder(AAE)[6] use GAN to fit posterior and prior, making the training process slow and complex. Unlike GMVAE and AAE, Variational Deep Embedding(VaDE)[7] stick to the traditional representation of distributions and combine GMM with two stage training process, which deepen the whole model and make it perform better with high robustness and fast training speeds.

D. *Auto-Encoder with SVM*

AE(Auto-encoder) has a deep network structure which allows the original input image to be trans- formed into a latent space non-linearly. Deep Auto-Encoder have proven useful for dimensionality reduction and image denoising recently. On the image classification task, many methods have be proposed, mostly people use a CNN-like network to extract features then classify image based on these features. We don't know how networks extract features or how related or seperable these features are since neural network is a black box, but AE has second network to evaluate how good these features can represent original image, so it should be more explainable on selected features.

In our project, since a high-resolution image dataset was chosen to fit model, CAE(Convolutional Auto- encoder) should be the best choice. Also, adding a FC layer[8] can make final latent dimentionality controllable. Finally, using SVM to seperate these features. The high accuracy in the experiment shows that our model have strong ability to handle such image classification task.

Contributions This paper mainly contributes on:

- Compare the accuracy difference between ResNet, VGG16 and VGG16-SVM.
- Test the performance of VGG16 and SVM in big data sets after combining.
- Improve the feature extraction ability for high-resolution images by combining ResNet with two models.
- Compare the performance and robustness of GMVAE and VaDE under fine tuning with same high resolution AFHQ dataset.
- Analysis the shortage of VAE based methods on dealing with more complex distributions.
- Testing how well a shallow CAE with only few layers can work on a high-resolution dataset.
- Make some experiments on robustness when AE handling dirty data.

III. Experiments

This section evaluate the performance of four approaches above on a high-resolution datasets of animal faces.

A. *Dataset Description*

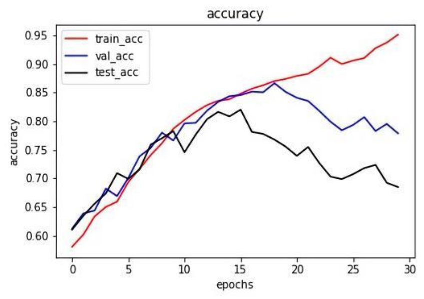
The dataset is known as Animal Faces-HQ (AFHQ), which consists of 16,130 high-quality images at 512×512 resolution. There are three domains of classes, each providing about 5000 images. By having multiple (three) domains and diverse images of various breeds per each domain, AFHQ sets a challenging image-to-image translation problem. The classes are: Cat; Dog; Wildlife. For unsupervised task, The project extract another five types of animals from Wildlife and reduced each class to about 500 images. The training set, validation set and test set are in proportion of 80%, 10%, 10% Transformation is to resize each images to 224x224, loaded in to a range of [0, 1] and then normalized using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225].

B. *Resnet-50*

This part adopt Resnet which consists 50 layers. Set the optimizer to Adam.

1)Initial round with transfer learning

First the work train the model with no transfer learning methods and initiate it with parameters with randomly chosen from a Kaiming uniform distribution. But it do not work out well.(only 80% in training set, 60% in validation set and test set). To gain a higher accuracy in 30 epochs, this work adopt transfer learning methods. This part use the pre-train model and load its parameters into our model. Then set the hyperparameters: using Adam as the optimizer and setting learning rate to 0.0001, range of β is (0.9, 0.999), epoches are 30 and batch sizes are 16.The results of this experiment is shown as Figure 1.



a) Accuracy with initial setting with transfer learning
Figure 1: Initial setting with transfer learning

Clearly This result gain a lot in the accuracy and loss. Accuracy surges to 95% in training set and 80% in validation sets. And the loss decreases from 0.5 to 0.2 in training set and from about 0.65 to 0.5 in validation set. But it seems that the model is not good enough, it has a severe overfitting problem. So this part tune the hyperperameters and try with different combinations of hyperparameters.

2)The best results of tuning in the hyperparemeters on top of transfer learning

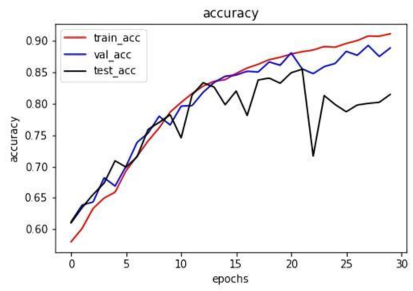
First the work do the tuning using optimizer Stochastic Gradient Descent:

- 5 groups of learning rate: (0.01, 0.005, 0.001, 0.0005, 0.0001)
- 3 groups of momentum: (0.8, 0.85, 0.9)

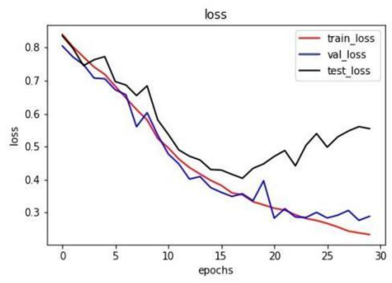
- 3 groups of weight decaying rate: (0.01, 0.001, 0.0001) Second the work do the tuning using optimizer Adam:
- 5 groups of learning rate: (0.01, 0.005, 0.001, 0.0005, 0.0001)
- 2 groups of β : (0.8, 0.9), (0.9, 0.999)
- 3 groups of weight decaying rate: (0.01, 0.001, 0.0001)

after trying all the combination of hyperparameters in Adam and SGD, the result get the best combina- tion(critiria is the highest accuracy in test set) in two optimizers are:

- SGD: learning rate = 0.0001, momentum = 0.9, weight decaying = 0.0001
- Adam: learning rate = 0.0001, betas = (0.8, 0.999) weight decaying = 0.0001



(a) accuracy of Adam:lr = 0.0001, betas= (0.8, 0.999) weight decaying = 0.0001loss of SGD:lr = 0.0001,momentum= 0.9, weight decaying = 0.0001



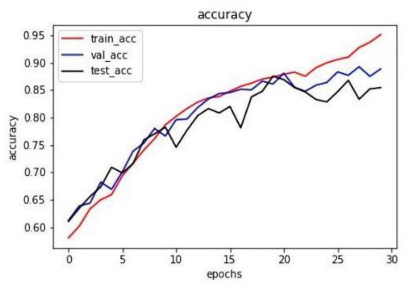
(b) loss of Adam:lr = 0.0001, betas = (0.8, 0.999) weight decaying = 0.0001
Figure 2: best combination of Adam

The results of two seperate best combination from two optmizers respectively are shown as Figure 2. From the figures that after the hyperparameters tuning, there is a improvement in the test accuracy and loss. Since best result using Adam outweights the SGD, we just show the result of Adam. At this time the models using Adam seems to perform pretty well in the training set and validation set but there still exist the problem of overfitting in loss when test on the test set. What's more, based on observing the curve of accuracy, the work have a conclusion that if this project adopt early stopping at epoch 20, there is about 82% accuracy in test set. To deal with the overfitting problem, this part choose some common methods.

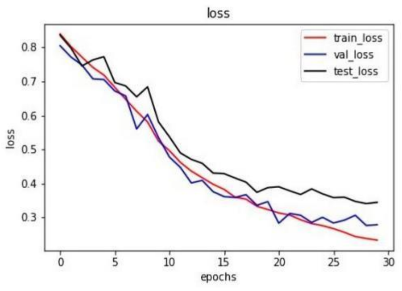
3)Methods of fixing overfitting and the results

Since from the former section the project conclude that the Adam optimizer is better than SGD. So this section use the measures which could cope with overfitting and redo all the combination and try to find the best results. This section eployees the horizontal flip, random vertical flip, random rotation to enlarge the data sets and add the drop out function in the fully connecting layers. Besides, the weight decaying indicates that there exists L_2 norm on top of our cross entropy loss function. (η is the penalty parameters) And the result find out the best combination(criteria is the highest number the section get in test accuracy) of Adam increases slightly, the accuracy reaches about 87% if we adopt early stopping at epoch 19.

The accuracy and loss in the training epochs of the best combination are shown in Figure 3.



(a) Accuracy with initial setting with transfer learning



(b) Loss with initial setting with transfer learning
Figure 3: Initial setting with transfer learning

4)Conclusion of experiment on Resnet-50

From the above, we can see that the measures we take to cope with overfitting gain a lot of improve- ment in both accuracy and loss in the test set. In addition, if early stopping is adopt in epochs 20, the accuracy can reach 86%, which is the highest we can get. Also, loss reduce from 0.5 to 0.35 after adopting these methods(data augmentation, drop-out, L_2 regularization). To sum up, the best accuracy of Resnet-50 can get is 87% and 0.35 in loss(cross entropy loss).

C. VGG16

The section set epochs to 100 with a early stop when the validation accuracy has passed five epochs and is not increasing, and set the opitmizer to Adam and the project will add data enhancements to all VGG16 tests.

1)Initial round

First the work do not use transfer learning and try to see the learing ability of VGG16. This section use Adam as the optimizer and setting batch size to 64, learning rate to 0.01, decay to 1e-3, and activation to linear. And the results are followed.

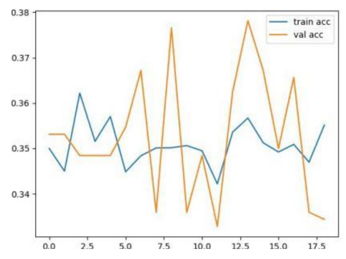


Figure 4: Accuracy with initial setting

Figure 4 shows that cannot gain a high accuracy in three sets(36% in training set, 35% to 36% in validation set and 35.1% in test set). All the losses the result get are nan, so this section don't have a graph of loss

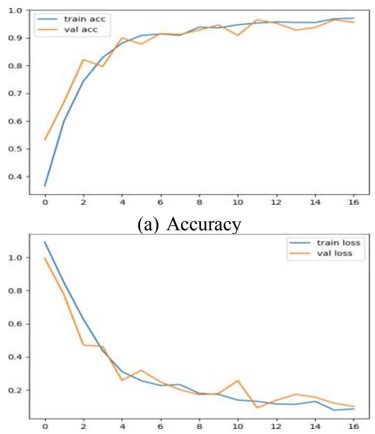
2)The best results of tuning in the hyperparameters on top of VGG16

This section do the tuning using optimizer Adam and L2 normalization:

- 3 group of batch size: (6, 32, 64)
- 4 group of learning rate: (0.1, 0.01, 0.001, 0.0001)
- 3 group of decay: (1e-2, 1e-3, 1e-6)
- 4 group of L2: (0.1, 0.01, 0.001, 0.0001)
- 4 group of activation:(relu, tanh, sigmod, linear)

after trying all combination of hyperparameters in Adam, the result get the best combination in Adam is:

- Adam:Batch size = 64, learning rate = 0.0001, decay = 1e-6, L2 = 0.0001, activation = relu



(a) Accuracy

(b) Loss

Figure 5: Best hyperparameters in Adam

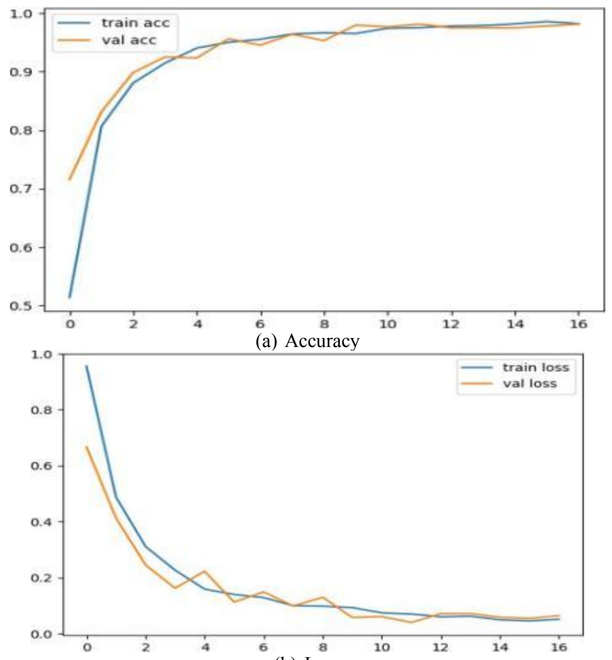
Table 1: VGG test Accuracy

VGG test Accuracy	Without transfer	With transfer	Best results
Test Acc	0.351	0.903	0.983
Test loss	Nan	0.134	0.064

Figure 5 shows that the train accuracy and validation accuracy reach to 96%. However, from the table 1 the test accuracy gain 90.3%. So this section need to deal with the overfitting for VGG16.

3)Method of fixing overfitting and the result

This part got the best hyperparameter combination from the previous section, but there is an overfitting problem. So this section can deal with over-fitting measures, redo all the combinations, and try to find the best results. This section used the most common method to add dropout after the fully connected layer, and continue to use the L2 normalization in the previous section to help solve the overfitting problem. The project have found the best combination of Adam in Figure 6, which has changed slightly from before: Adam:Batch size = 64, learning rate = 0.0001, decay = 1e-6, L2 = 0.0001, activation = relu, dropout = 0.1.



(a) Accuracy

(b) Loss

Figure 6: Best hyperparameters in Adam with dropout

Through table 1 the test accuracy is 98.3%, so when dropout is 0.1. So, the results can fix the overfitting problem and gain a high accuracy.

4)VGG16 extract feature then use SVM classify

This section train vgg16 and save it as a .5h file. Extract the penultimate fully connected layer, then needing from the file as features, and input the features and labels. Then put those into SVM for classification. This part use the gridsearchCV method to tune the hyperparameters of the SVM.

- 6 group of C: (0.001, 0.01, 0.1, 1, 10, 100)
- 6 group of gamma: (0.001, 0.01, 0.1, 1, 10, 100)2 group of Kernel: (rbf, linear)

The result got the best hyperparameter combination through grid search. However, did not get a high accuracy rate for the output.

- ’C’: 1, ’gamma’: 0.001, ’kernel’: ’rbf’

It show on Table 2.

Table 2: VGG-svm Accuracy

Train	Validation	Test
37.29	35.09	35.07

5)Conclusion of experiment on VGG16

From the above shows that the measures the project have taken to deal with over-fitting have greatly improved the accuracy and loss of the test set. After this section adopted the dropout method, the loss dropped from 13.4% to 6.4%. Comparing our best hyperparameter combinations in VGG16 and VGG16-SVM, From the research can clearly see that VGG16-SVM does not perform well in this data set. In summary, the best accuracy rate obtained by the method based on VGG16 is 98.3%, and the loss is 6.4%.

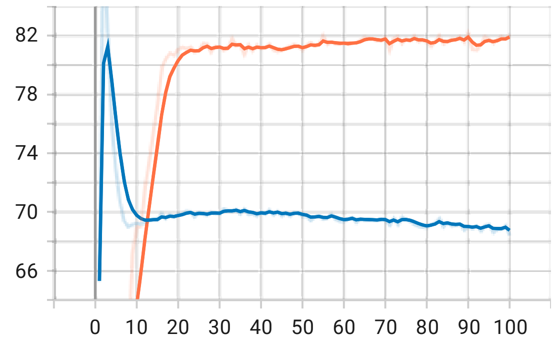
D. VAE based

In this section display the results of our experiments in order to understand VAE based methods better. This section is divided in three parts:

1. Evaluate the model with ResNet50 as feature extraction.
2. Study the stability of two models with different random seeds.
3. Visualized the latent space of one model.

Many experiments have proved the great clustering performance of VAE based methods on low- dimension inputs. VAE use shallow linear layers in the form of encoder and decoder to represent learning process of probability distribution, which gave the model abilities to extract latent features from huge dataset by forwards and backwards techniques. But it constrains the model to learn more complex distribution, for the linear layers lack of representative for images in nature. Experiments shows, if the project contain couple of CNNs into the training process, network would not converge. We believe it is due to the reparameterization trick between the encoder and decoder need meaningful and easy-to-control latent codes, together with other parameters learn from the autoencoder and GMM, to optimize on its objective function. CNN will bring much more parameters and reduce the interpretability of the model.

In conclusion, VAE can only deal with low-dimension datasets due to its shallow linear networks. It has advantage in training and converge speed. It will converge after 10 epochs as show in 10, and each epoch only require 20 seconds in general. But the results are highly depended on feature extraction stage. The final results here use pretrained ResNet50 to extract a 2048-dimension vector for each image first. If the section directly flatten the RGB image or use gray scale of it or includes CNN layers like ResNet into our model, it will only get accuracy slightly above the random results, which shows how much the network depends on the quality of input vector.


(a) GMVAE train ACC.

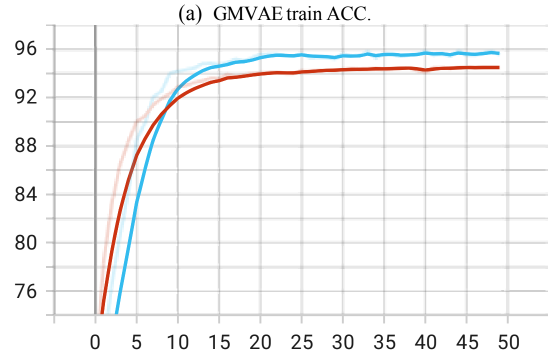
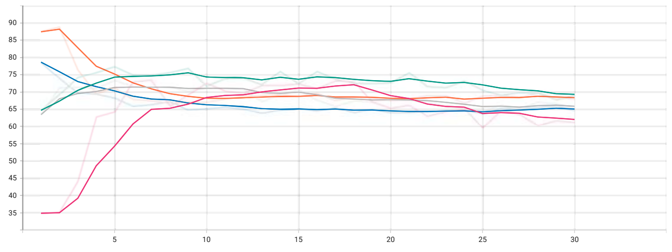
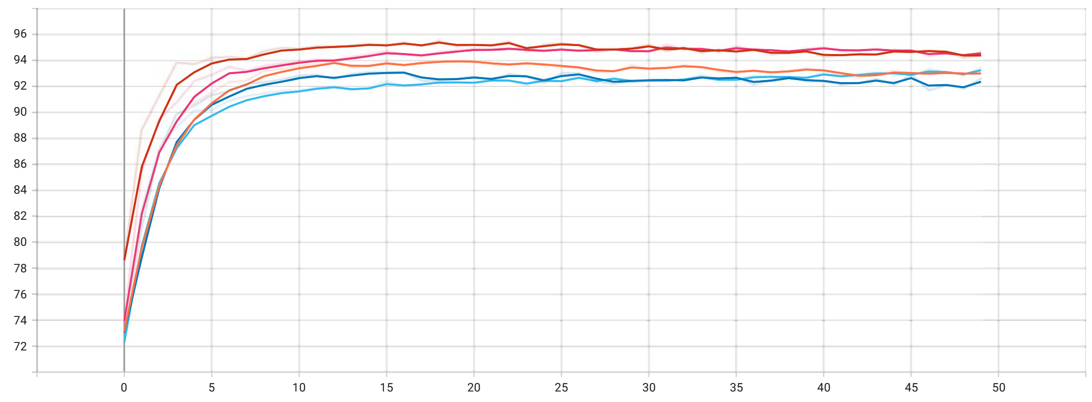

(b) VAE train ACC.

Figure 7: Train Accuracy of two model

Notice that the 3 class GMVAE has a high peak at about 3 epoch Figure 7(a), than full down quickly. At first, we thought it is due to over-fitting with large dataset (7 class did not show it). Later we find out it is due to the poor robustness of the model through the experiment show in Figure 8. We only show it with 3 class because 3 class datasets have more data than 7 class, GMVAE under 3 class will converge while 7 class will not. VaDE on the other hand is much more robust in both datasets.


(a) GMVAE 3 class val ACC.



(b) VAE 3 classval ACC.
Figure 8: Stability test of two model

At last, to demonstrate VaDE's representation power as an unsupervised model, we visualized the latent space of it in Figure 9. We reduce the dimensions of latent space from 10 to 3 by t-SNE[9]. What is interesting is that, also the 3 class results of wild class all colored in yellow, we can see it automatically gathered together into 5 balls, which is exactly the number of animal types in wild class.

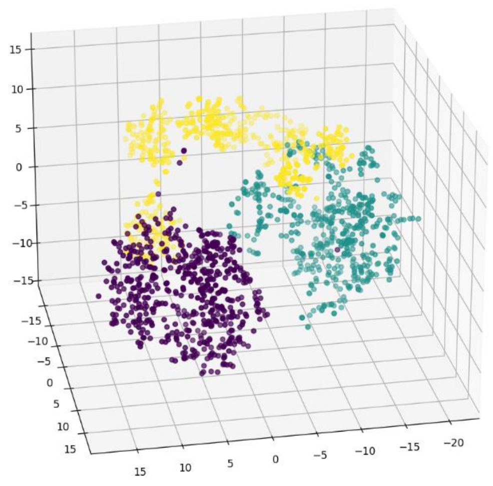


Figure 9: Visualization of VaDE latent space

In addition, the learning rate we choose is 1e-4, hidden size is 10, 20 batch size for 7 class, 64 batch size for 3 class. The typical results of test accuracy (without consider of stability) are showed in Table 3.

Table 3: Test Accuracy

Datasets	GMVAE	VaDE
3 Class	69.44	96.67
7 Class	82.78	94.19

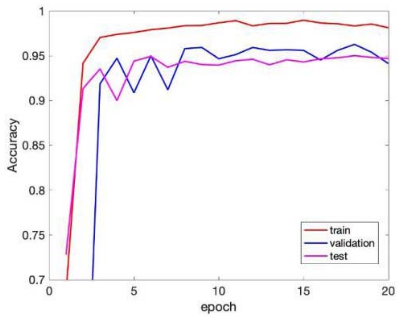
E. Auto-Encoder

We implemented our method In Python with TensorFlow 2.0 and evaluate it on a high-resolution dataset. The result was compared with the result after data augmentation. Also noise was added in dataset to see how robust our model is. A dataset was created with dirty data of 5, 10, 15 and 20 percent. The result shows our model has a certain degree of robustness when handling dirty data under 10 percent.

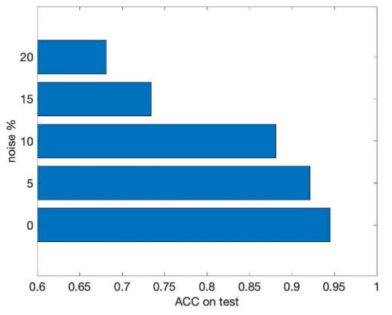
1)Training strategy

As illustrated in Figure 10, we first pre-train the deep auto-encoder without the FC layer on all the data we have. We then use the pre-trained parameters to initialize the encoder and decoder layers of our network. After this, in the fine-tuning stage, we use big batch to train the complete model. Specifically, we use Adam[10], an adaptive momentum based gradient descent method to minimize the loss, where we set several learning rates to find best result.Once the model is trained, we use this auto-encoder to reduce dimentionality of inputs, then we apply svm on low-dimention data, in each epoch we never change hyperparameter of svm, we set c as 1.0 and kernel as 'rbf'.

In Adam fine tuning we test with 5 groups of leanring rates: 0.0001, 0.0005, 0.001, 0.005, 0.01 and 2 groups of momentum: [0.8, 0.999], [0.9, 0.999]. When leanrning rate is large the result can't converge, when learning rate is small, the speed getting very slow. So finally in our work the learning rate was setted as 0.05 and beta momentum as [0.8, 0.999]. Our best accuracy on test set is 0.9453.



(a) Result of origin data



(b) Accracy with noise

Table 4: Results with noise

Data set	train	test	validation
origin	0.9891	0.9453	0.9560
augmentation	0.9909	0.9570	0.9647
Gaussian noise 5%	0.9815	0.9212	0.9355
Gaussian noise 10%	0.9316	0.8812	0.8943
Gaussian noise 20%	0.7816	0.6812	0.7015

Figure 10: Evaluation of the model. In figure 10, the result shows how our model converge and how robust

when facing noise. Table 4 shows exact result of robustness test.

Since there is a FC layer, though convolution is not sensitive to image flipping or rotation, they can effect a FC layer, so we apply data augmentation before training and the final result got a little better.

After data augmentation, the result had an improvement. In our work noise was also added on data, testing robustness of our model. In model we use Gaussian noise, and dirty only part of our data. In experiment the result showed when the percentage Gaussian noise added on our data set was under 10, the accuracy was still high and accuracy didn't decrease very quick. When the percentage is higher, we had our result decrease very badly. Final result of this experiment in Table 4 showed our model remain robust when facing data set with low percentage of dirty data.

IV. Conclusion

The project applied deep learning methods based on ResNet and VGG and auto-encoder with SVM to classify AFHQ. Then the project uses VAE based methods to cluster the same dataset. Our results are state-of-the-art on this dataset which proved those modules has good abilities to deal with high resolution images. We believe it can greatly improve the efficiency of wild life monitoring and provide reliable data support for the formulation of wildlife protection strategies.

Reference

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 2016.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[3] Yichuan Tang. Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239, 2013.

[4] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. Journal of the American statistical Association, 112(518):859–877, 2017.

[5] Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture

[6] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adver- sarial autoencoders. arXiv preprint arXiv:1511.05644, 2015.

[7] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. arXiv preprint arXiv:1611.05148, 2016.

[8] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. Deep subspace clustering networks, 2017.

[9] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of machine learning research, 9(11), 2008.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2017.