Math 6380p: Adv. Top. Deep Learning **28 Nov, 2018**

Final Projects.

Instructor: Yuan Yao *Due: 00:00am Monday 17 Dec, 2018*

1 Project Requirement

This project as a warm-up aims to explore feature extractions using existing networks, such as pre-trained deep neural networks and scattering nets, in image classifications with traditional machine learning methods.

1. Pick up ONE (or more if you like) favourite dataset below to work. If you would like to work on a different problem outside the candidates we proposed, please email course instructor about your proposal.

2. Team work: we encourage you to form small team, up to FOUR persons per group, to work on the same problem. Each team just submit ONE report, *with a clear remark on each person's contribution*. The report can be in the format of either Python (Jupyter) Notebooks with a detailed documentation (preferred format), a *technical report within 8 pages*, e.g. NIPS conference style

   `https://nips.cc/Conferences/2016/PaperInformation/StyleFiles`

   or of a *poster*, e.g.

   `https://github.com/yuany-pku/2017_math6380/blob/master/project1/DongLoXia_poster.pptx`

3. In the report, show your proposed scientific questions to explore and main results with a careful analysis supporting the results toward answering your problems. Remember: scientific analysis and reasoning are more important than merely the performance tables. Separate source codes may be submitted through email as a zip file, GitHub link, or as an appendix if it is not large.

4. Submit your report by email or paper version no later than the deadline, to the following address (deeplearning.math@gmail.com) with Title: Math 6380P: Project 3.

2 New Challenges Self-proposed by Classmates

2.1 Smartphones glass defects detection and pixel-wise classification

This project is about smartphones glass defects detection and pixel-wise classification. The purpose is to analyze the response of Scatnet, Resnet, VGG, and DCF-Based Networks in terms of accuracy and speed.

Robotics and Multi-Perception Lab (ram-lab.com) has created a glass defects datasets which have 80-100 images of smartphones glasses with defects such as Scratch, Pit, Crack, Chip, Dirt. Each image is taken from the 16K line camera. Size of each image is about 400-500MBs and the total size is about 21GBs. The main task to detect the defects and pixel-wise classification of these defects. The minimum defect to detect and classify should be about $\sim$10 micrometers. Each pixel is about $\sim$0.5-micron size.

You can request dataset from (`mumbhutta@connect.ust.hk`) if interested. This challenge is proposed and pursued by Muhammad Usman Maqbool BHUTTA (`mumbhutta@connect.ust.hk`), Yuan LAN (`ylanaa@connect.ust.hk`), and Candi ZHENG (`czhengac@connect.ust.hk`).

Figure 1: A sample of glass defect that contains all the four types, scatches, cracks, chip, and dirt.

2.2 EmoContext: A Shared Task on Contextual Emotion Detection in Text

We routinely experience emotions such as happiness, anger, sadness etc. As humans, on reading "Why don't you ever text me!", we can either interpret it as a sad or an angry emotion in absence of context; and the same ambiguity exists for machines as well. Lackness of facial expressions and voice modulations makes detecting emotions in text a challenging problem. However, as we increasingly communicate using text messaging applications and digital agents, contextual emotion

detection in text is gaining importance to provide emotionally aware responses to users. The shared task aims to bring more research to the problem of contextual emotion detection in text.

In this task, you are given a textual dialogue, i.e. a user utterance along with two turns of context. One has to classify the emotion of user utterance as one of the emotion classes: Happy, Sad, Angry or Others. The training data set contain 15K records for emotion classes, i.e., Happy, Sad and Angry combined, and contains 15K records not belonging to any of the aforementioned emotion classes. This challenge has been proposed as SemEval 2019 task (`http://alt.qcri.org/semeval2019/`), there are already more than 100 submission in the scorer system. More information can be found in `https://www.humanizing-ai.com/emocontext.html` and some baseline code in `https://github.com/SenticNet/conv-emotion/blob/master/bc-LSTM/baseline.py`.

This challenge is proposed and pursued by Andrea MADOTTO (`amadotto@connect.ust.hk`), Genta Indra WINATA (`giwinata@connect.ust.hk`), Zhaojiang LIN (`zlinao@connect.ust.hk`), and Jay SHIN (`jay.shin@connect.ust.hk`) from CAiRE.

2.3 Exploring the Robustness of Neural Network

Despite its numerous successful applications, neural network, it is susceptible to natural noises or artificial attacks. Therefore, the robustness of neural network arouse our interests in this challenge.

Two particular tasks will be pursued here. On one hand, we will verify DCFNet's robustness against high-frequency noise. As proposed in the (https://arxiv.org/abs/1802.04145), DCFNet's structured filters should equip it the ability against high-frequency noise. Quantitative verification on this hypothesis will be studied. On the other hand, we will extend the term "noise" to adversarial examples, which is introduced in Lecture 22. In particular, we will try to reproduce the method proposed in (`https://arxiv.org/abs/1704.08847`), which try to improve robustness of networks by adding constraint on its Lipschitz constant. The Lipschitz constraint can be done for each layer by adding orthogonal constraint on the weight matrix (e.g. convolutional operators). The singular values of the weight matrix are constrained to be 1 and the Lipschitz constant of whole neural network is smaller than 1. It is hoped that smaller perturbation of input will not affect output too much.

One can use various attack methods to attack a network to see whether it is robust to adversarials. Several Attack method have been introduced in Lecture 22 including FGSM, PGD and CW etc. One can use MNIST or CIFAR10 to do this task with whatever network architecture that one wants to explore like VGG16 or ResNet18.

This challenge is proposed and pursued by Zhichao, HUANG (`zhuangbx@connect.ust.hk`) and Zhicong, LIANG(`zliangak@connect.ust.hk`).

2.4 Denoising in Cryo-EM Imaging Problem

2.4.1 Introduction

The Cryo-EM images are the projection images of frozen samples by the electron microscope, whose process may introduce large and irregular noise. In order to classify the different structure of elements and reconstruct the biological structure well, denoising is becoming a very important process. This challenge is proposed and pursued by Hanlin GU (`hguaf@connect.ust.hk`) and his peers in Prof. Xuhui HUANG's Lab at HKUST.

2.4.2 Data

The data is mainly a special structure of bacterial RNA Polymerase which includes 50000 clean images of size 128*128 (`http://143.89.53.65/shared_folder/hanlin_project/`) (need to `import mrcfile` in python to read data), you can add any types of noise in different SNR (signal noise ratio) in clean images using python function (or any functions you like):

```
skimage.util.random_noise(image, mode=, seed=None, clip=True, **kwargs)
```

2.4.3 Process

Try traditional denoising methods, for example transform domain methods (BM3D), dictionary learning methods (KSVD) and so on. Or you can try deep learning methods like VAE or GAN to denoise, the following is one implementation using GAN by Cianfrocco's Lab in the University of Michigan:

```
https://github.com/cianfrocco-lab/GAN-for-Cryo-EM-image-denoising
```

Can you try to create your own new network architecture to denoise Cryo-EM images? After denoising, you can set an criterion to evaluate the performance, a direct method is calculate the mean square error between clean images and noisy denoised images. A good try is to compare different methods in different SNR levels to understand the performance of different denoising methods well.

2.4.4 Further exploration

There are several challenges in Cryo-EM denoising: (A) Can the method denoise well when SNR is low? (B) Is your method also suitable for experimental real world data? Some real world experimental data can be downloaded in the website `https://www.ebi.ac.uk/pdbe/emdb/empiar/`. For experimental data, it it worth to mention that some references may be derived by softwares: CRYOSPARC `https://cryosparc.com/docs/tutorials/`, and you can also using this software to reconstruct 3D structure.

3 Reinforcement Learning for Image Classification with Recurrent Attention Models

This task basically required you to reproduce some key result of Recurrent Attention Models, proposed by Mnih et al. (2014). (See `https://arxiv.org/abs/1406.6247`)

3.1 Cluttered MINIST

As you've known, the original MNIST dataset is a canonical dataset used for illustration of deep learning, where a simple multi-layer perceptron could get a very high accuracy. Therefore the original MNIST is augmented with additional noise and distortion in order to make the problem more challenging and closer towards real-world problems.

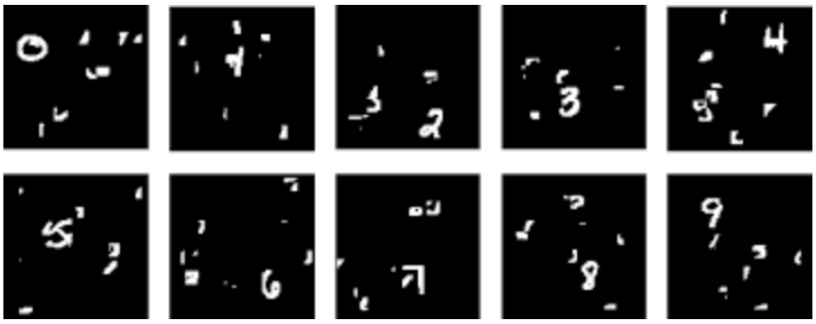

Figure 2: Cluttered MNIST data sample.

The dataset can be downloaded from `https://github.com/deepmind/mnist-cluttered`.

3.2 Raphael Drawings

For those who are interested in exploring authentication of Raphael drawings, you are encouraged to use the Raphael dataset:

`https://drive.google.com/folderview?id=0B-yDtwSjhaSCZ2FqN3AxQ3NJNTA&usp=sharing`

that will be discussed in more detail later.

3.3 Recurrent Attention Models (RAMs)

The efficiency of human eyes when looking at images lies in attentions – human do not process all the input information but instead use attention to select partial and important information for perception. This is particularly important for computer vision when the input images are too big to get fully fed into a convolutional neural networks. To overcome this hurdle, the general idea of

RAM is to model human attention using recurrent neural networks by dynamically restricting on interesting parts of the image where one can get most of information, followed by reinforcement learning with rewards toward minimizing misclassification errors.

You're suggested to implement the following networks and train the networks on the above Cluttered MINIST dataset.

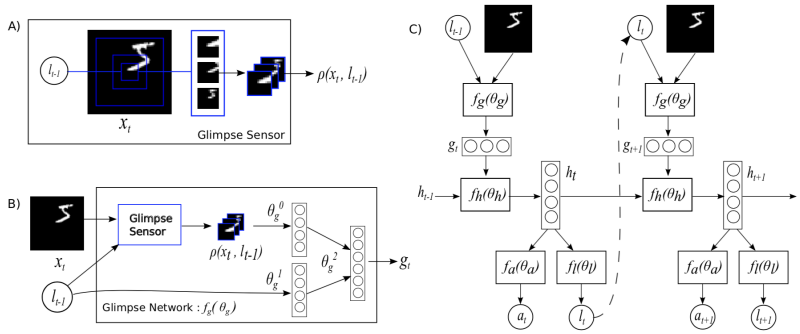

Figure 3: Diagram of RAM.

- Glimpse Sensor: Glimpse Sensor takes a full-sized image and a location, outputs the *retina-like* representation $\rho(x_t, l_{t-1})$ of the image x_t around the given location l_{t-1}, which contains multiple resolution patches.

- Glimpse Network: takes as the inputs the retina representation $\rho(x_t, l_{t-1})$ and glimpse location l_{t-1}, and maps them into a hidden space using independent linear layers parameterized by θ_g^0 and θ_g^1 respectively using rectified units followed by another linear layer θ_g^2 to combine the information from both components. Finally it outputs a glimpse representation g_t.

- Recurrent Neural Network (RNN): Overall, the model is an RNN. The core network takes as the input the glimpse representation g_t at each step and history internal state h_{t-1}, then outputs a transition to a new state h_t, which is then mapped to action a_t by an action network $f_a(\theta_a)$ and a new location l_t by a location network $f_l(\theta_t)$. The location is to give an attention at next step, while the action, for image classification, gives a prediction based on current informations. The prediction result, then, is used to generate the reward point, which is used to train these networks using Reinforcement Learning.

- Loss Function: The reward could be based on classification accuracy (e.g. in Minh (2018) $r_t = 1$ for correct classification and $r_t = 0$ otherwise). In reinforcement learning, the loss can be finite-sum reward (in Minh (2018)) or discounted infinite reward. Cross-entropy loss for prediction at each time step is an alternative choice (which is not emphasized in the origin paper but added in the implementation given by Kevin below). It's also interested to figure out the difference function between these two loss and whether it's a good idea to use their combination.

To evaluate your results, it is expected to see improved misclassification error compared against feedforward CNN without RAMs. Besides, you're encouraged to visualize the glimpse to see how the attention works for classification.

3.4 More Reference

- A PyTorch implementation of RAM by Kevin can be found here:
  `https://github.com/kevinzakka/recurrent-visual-attention`.
- For reinforcement Learning, David Silver's course web could be found here
  `http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html`
  and Ruslan Satakhutdinov's course web at CMU is
  `http://www.cs.cmu.edu/~rsalakhu/10703/`

4 Challenges from Project 2

The following proposes three candidates and you are welcome to propose your own research project. Previous challenges are collected in the end and you may pursue a deeper exploration.

4.1 Nexperia Kaggle in-class Contest

Nexperia (`https://www.nexperia.com/`) is one of the biggest Semi-conductor company in the world. They will produce billions of semi-conductors every year. But unfortunately, they are facing a hard problem now which is the yield rate of the semi-conductors. However they have lots of data and hope that the yield rate could be greatly improved by the hot deep learning technics now. So with the data they provide to us, we lunch this in-class Kaggle contest which tries to use various machine learning and deep learning methods to solve this real world problem.
Because this is the first Nexperia image classification contest, we set only 2 classes, one for bad semi-conductor and another for good. The aim of this simplified contest is to predict the type of each semi-conductor based on the image. And we provide 30K and 3000 images for training and testing respectively.

Checking the following kaggle website for more details.

- Kaggle: `https://www.kaggle.com/c/semi-conductor-image-classification-1`

To participate the contest, you need to login your Kaggle account first, then open the following invitation link and accept the Kaggle contest rule to download the data:

`https://www.kaggle.com/t/7002fff75f2c422cb34068731971afcd`

In the future, we may consider to lunch another more complex in-class Kaggle contest. But this project is not limited to classification, you can do various explorations such as visualization 5 and abnormal outlier detection.

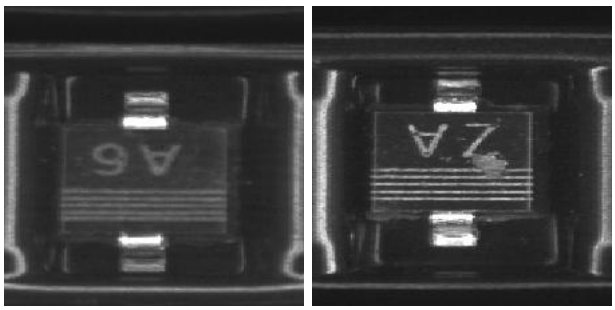

Figure 4: Sample Semi-conductor Image. Left: good Right: bad

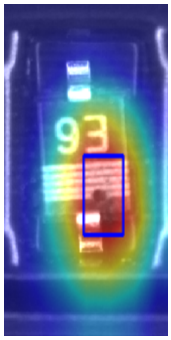

Figure 5: Sample Semi-conductor Visualization.

4.2 DCF-Net Exploration

This challenge is to implement the DCF-Net in image classification tasks, defined by Xiuyuan Cheng et al. in the following paper

Qiang Qiu, Xiuyuan Cheng, Robert Calderbank, Guillermo Sapiro, *DCFNet: Deep Neural Network with Decomposed Convolutional Filters*, ICML 2018. `arXiv:1802.04145.`

Currently there are two implementations,

- Matlab: `https://github.com/xycheng/DCFNet`
- Pytorch: `https://github.com/ZeWang95/DCFNet-Pytorch`

You may train a DCF-Net, e.g. DCF-Net-VGG16 or DCF-Net-ResNet18, on your favorite datasets (e.g. MNIST/Fashion-MNIST/Cifar10), and compare them against pretrained VGG16 and ResNet18 etc. For example, Table 4 of the DCF-Net paper, shows comparison between imagenet-vgg-verydeep-16 and DCFNet based VGG 16. You may either reproduce such an experiment and/or explore new models with new datasets.

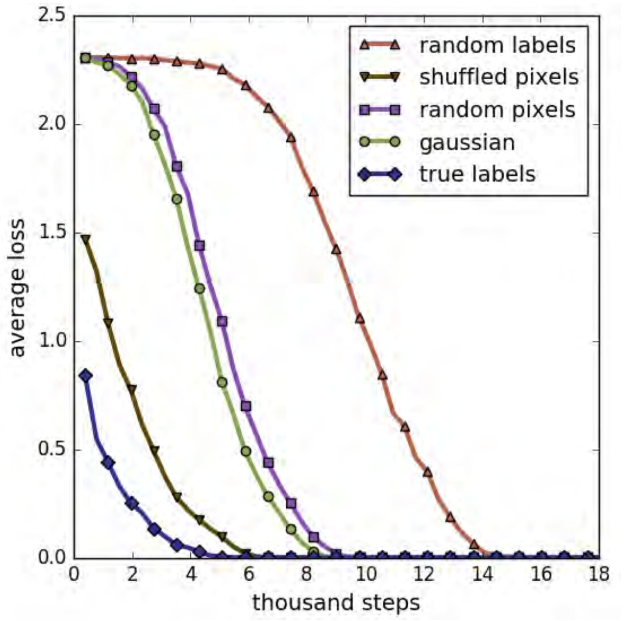

Figure 6: Overparametric models achieve zero *training error* (or near zero *training loss*) as SGD epochs grow, in standard and randomized experiments.

4.3 Reproducible Training of CNNs

The following best award paper in ICLR 2017,

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals, Understanding deep learning requires rethinking generalization. `https://arxiv.org/abs/1611.03530`

received lots of attention recently. Reproducibility is indispensable for good research. Can you reproduce some of their key experiments by yourself? The following are for examples.

1. Achieve ZERO training error in standard and randomized experiments. As shown in Figure 6, you need to train some CNNs (e.g. ResNet, over-parametric) with Cifar10 dataset, where the labels are true or randomly permuted, and the pixels are original or random (shuffled, noise, etc.), toward zero training error (misclassification error) as epochs grow. During the training, you might turn on and off various regularization methods to see the effects. If you use loss functions such as cross-entropy or hinge, you may also plots the training loss with respect to the epochs.

2. Non-overfitting of test error and overfitting of test loss when model complexity grows. Train several CNNs (ResNet) of different number of parameters, stop your SGD at certain large enough epochs (e.g. 1000) or zero *training error (misclassification)* is reached. Then compare the *test (validation) error* or *test loss* as model complexity grows to see if you observe similar phenomenon in Figure 7: when *training error* becomes zero, *test error* (misclassification) does not overfit but *test loss* (e.g. cross-entropy, exponential) shows overfitting as model complexity grows. This is for reproducing experiments in the following paper:

Tomaso Poggio, K. Kawaguchi, Q. Liao, B. Miranda, L. Rosasco, X. Biox, J. Hidary, and H.

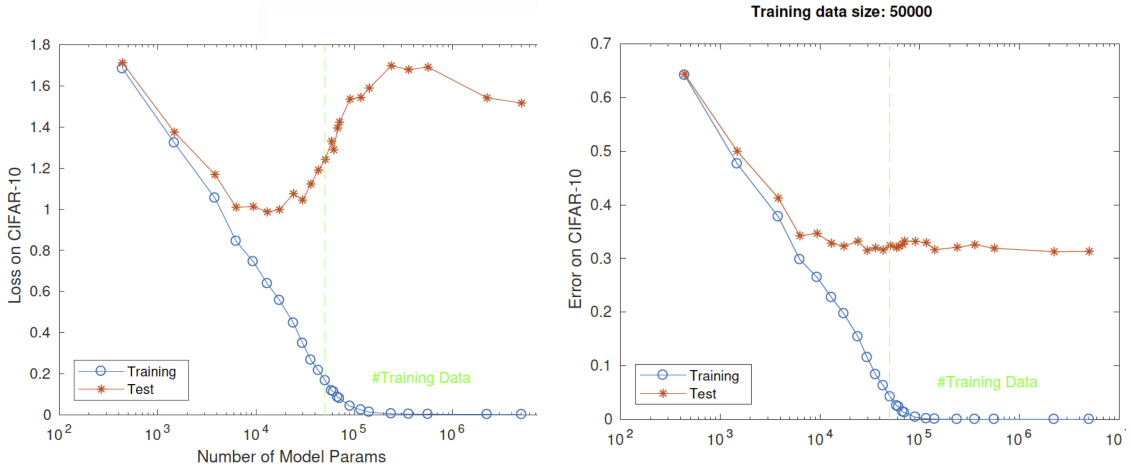

Figure 7: When *training error* becomes zero, *test error* (misclassification) does not increase (resistance to overfitting) but *test loss* (cross-entropy/hinge) increases showing overfitting as model complexity grows.

Mhaskar. Theory of Deep Learning III: the non-overfitting puzzle. Jan 30, 2018. `http://cbmm.mit.edu/publications/theory-deep-learning-iii-explaining-non-overfitting-puzzle`

3. Can you give an analysis on what might be the reasons for the phenomena you observed?

5 Challenges from Project 1

The basic challenge from Project 1 is

- Feature extraction by scattering net with known invariants;
- Feature extraction by pre-trained deep neural networks, e.g. VGG19, and resnet18, etc.;
- Visualize these features using classical unsupervised learning methods, e.g. PCA/MDS, Manifold Learning, t-SNE, etc.;
- Image classifications using traditional supervised learning methods based on the features extracted, e.g. LDA, logistic regression, SVM, random forests, etc.;
- *Train the last layer or fine-tune the deep neural networks in your choice;
- Compare the results you obtained and give your own analysis on explaining the phenomena.

Below are two candidate datasets. Challenge marked by * above is only optional.

5.1 MNIST dataset

Yann LeCun's website contains original MNIST dataset of 60,000 training images and 10,000 test images.

`http://yann.lecun.com/exdb/mnist/`

There are various ways to download and parse MNIST files. For example, Python users may refer to the following website:

`https://github.com/datapythonista/mnist`

or MXNET tutorial on mnist

`https://mxnet.incubator.apache.org/tutorials/python/mnist.html`

5.2 Fashion-MNIST dataset

Zalando's Fashion-MNIST dataset of 60,000 training images and 10,000 test images, of size 28-by-28 in grayscale.

`https://github.com/zalandoresearch/fashion-mnist`

As a reference, here is Jason Wu, Peng Xu, and Nayeon Lee's exploration on the dataset in project 1:

`https://deeplearning-math.github.io/slides/Project1_WuXuLee.pdf`

5.3 Cifar10 dataset

The Cifar10 dataset consists of 60,000 color images of size 32x32x3 in 10 classes, with 6000 images per class. It can be found at

`https://www.cs.toronto.edu/~kriz/cifar.html`

5.4 Identification of Raphael's paintings from the forgeries

The following data, provided by Prof. Yang WANG from HKUST,

`https://drive.google.com/folderview?id=0B-yDtwSjhaSCZ2FqN3AxQ3NJNTA&usp=sharing`

contains a 28 digital paintings of Raphael or forgeries. Note that there are both jpeg and tiff files, so be careful with the bit depth in digitization. The following file

`https://docs.google.com/document/d/1tMaaSIrYwNFZZ2cEJdx1DfFscIfERd5Dp2U7K1ekjTI/edit`

contains the labels of such paintings, which are

1 Maybe Raphael - Disputed

2 Raphael

3 Raphael

4 Raphael

5 Raphael

6 Raphael

7 Maybe Raphael - Disputed

8 Raphael

9 Raphael

10 Maybe Raphael - Disputed

11 Not Raphael

12 Not Raphael

13 Not Raphael

14 Not Raphael

15 Not Raphael

16 Not Raphael

17 Not Raphael

18 Not Raphael

19 Not Raphael

20 My Drawing (Raphael?)

21 Raphael

22 Raphael

23 Maybe Raphael - Disputed

24 Raphael

25 Maybe Raphael - Disputed

26 Maybe Raphael - Disputed

27 Raphael

28 Raphael

There are some pictures whose names are ended with alphabet like A's, which are irrelevant for the project.

The challenge of Raphael dataset is: can you exploit the known Raphael vs. Not Raphael data to predict the identity of those 6 disputed paintings (maybe Raphael)? Textures in these drawings may disclose the behaviour movements of artist in his work. One preliminary study in this project can be: *take all the known Raphael and Non-Raphael drawings and use leave-one-out test to predict the identity of the left out image; you may break the images into many small patches and use the known identity as its class.*

The following student poster reports are good explorations

1) Hanlin GU, Yifei HUANG, and Jiaze SUN: `https://github.com/yuany-pku/2017_CSIC5011/blob/master/project3/05.GuHuangSun_poster.pdf`

2) Jianhui ZHANG, Hongming ZHANG, Weizhi ZHU, and Min FAN: `https://deeplearning-math.github.io/slides/Project1_ZhangZhangZhuFan.pdf`,

3) Wei HU, Yuqi ZHAO, Rougang YE, and Ruijian HAN: `https://deeplearning-math.github.io/slides/Project1_HuZhaoYeHan.pdf`.

The following papers by Haixia Liu et al. study art authentication using geometric tight frames and scattering transform, respectively, which might be useful reference for you:

`http://dx.doi.org/10.1016/j.acha.2015.11.005`

`https://www.sciencedirect.com/science/article/pii/S0165168418301105`