# Preferred Networks Internship Screening 2018 Coding task Chainer Application field

- In this task you will write some code to perform image processing using Chainer
- You may use Chainer, NumPy/CuPy, OpenCv as libraries in addition to the Python standard libraries.
- Use the latest version (4.0.0) of Chainer.
- In opencv, though there are both BGR-based and RGB-based implementations, please use the BGR-based one.
- Do not share or discuss any details of this coding task with anyone.
- Please tackle the task by yourself. Do not share or discuss this coding task with anyone including other applicants. If we find an evidence of leakage, the applicant will be disqualified. If one applicant allows another applicant to copy the answer, both applicants will be disqualified.
- We expect you to spend up to two days for this task. You can submit your work without solving all of the problems. Please do your best without neglecting your coursework.

## Submission materials

- Problem 1. Source code and trained model (saved by serializers.save_npz). Please specify the version of Python that you used (for example, 2.7, 3.6 etc.).
- Problem 2. Source code, trained model (saved by serializers.save_npz), and a report document file on (3) (pdf / word / txt)

## Evaluation

It is desirable that your submission satisfies the following. (These are not mandatory. Your submission does not need to satisfy all of them if you are too busy.)

- The source code is readable for other programmers.
- There is an appropriate amount of unit tests and/or verification code to ensure the source code is correct.
- It is easy to reproduce the experimental results.
- The submitted report is concise and easy to follow.

## How to submit

Create a zip archive with all of the submission materials and submit it on this form. Due date is Monday, May 14th, 2018, 23:59 JST.

## Inquiry

If you have any question on this problem description, please contact us at intern2018@preferred.jp. An updated version will be shared with all applicants if any change is made. Note that we cannot comment on the approach or give hints to the solution.

## Problem 1

Some times we preprocess images before training converting with a filter etc. Such filter itself can also be made by learning method. The advantage of the learning-based filter is that it can make complicated filters if you have enough data, and that it itself can be differentiated. In the following, we train a learning based filter which converts image from RGB color space to Lab color space.

- (1) Define a three layer neural network model using 1D convolution and ReLu function for mapping RGB -> Lab color space. Since opencv has a function for converting RGB color into the Lab color space, define a DataSet class that outputs a pair of data before and after conversion. Train the model using Trainer and the defined Dataset.
- (2) We want to evaluate the conversion accuracy for the model during training. Calculate the mean squared error with the target value for the entire RGB color space and define an Evaluator that outputs the averaged error and the worst error. Also, when you actually evaluate the model by loading the model trained in (1), make it possible to run the evaluator every N iterations (N is a constant) and check the operation. (If the checking is performed in the entire RGB space and the execution time could be too long, it is allowed to evaluate only on a part of image)
- (3) Train another network model that performs the inverse conversion from Lab to RGB color space and check the trained model using the evaluator.

## Problem 2

Image classification, object detection, segmentation, etc. can be considered as application examples of image recognition using deep learning. In Chainer, pre-trained models such as VGG and ResNet are available. In the following we apply these pre-trained models to specific problem setting.

- (1) Train a network that identifies several images taken from ImageNet (DATA/imagenet_data) and images from illustration (DATA/anim_face_data) by loading these pre-trained models and fine-tuning only the final output layer.

- (2) We want to detect imagenet_data as background and anim_face_data as foreground in pixel wise. To make sutch network, prepare synthesized dataset and train network using pre-trained network. First, make synthesized dataset which simply paste a illustration image into enlarged an ImageNet image at random positions. ( enlarged image size is fixed as twice of illustration image. ) Next define a Dataset class that returns a pair of generated image and mask image that shows the position of the pasted illustration image. Then create a network that predicts the mask image from the combined image. At that time, use a middle hidden layer of the model used in (1) (for example, the layer just before the fully-connected layer) In this problem, you may or may not assume that the shape of the pasted image is known to be a rectangle of fixed size.
- (3) Training using synthesized data as in (2) is important for creating a model for a certain purpose, but it may not have sufficient detection performance in real-world situations. When the model of (2) above does not give sufficient accuracy to the actual data, explain the expected reasons and what kinds of methods (except simply enlarge dataset) can be considered as countermeasures and explain some expected effects.