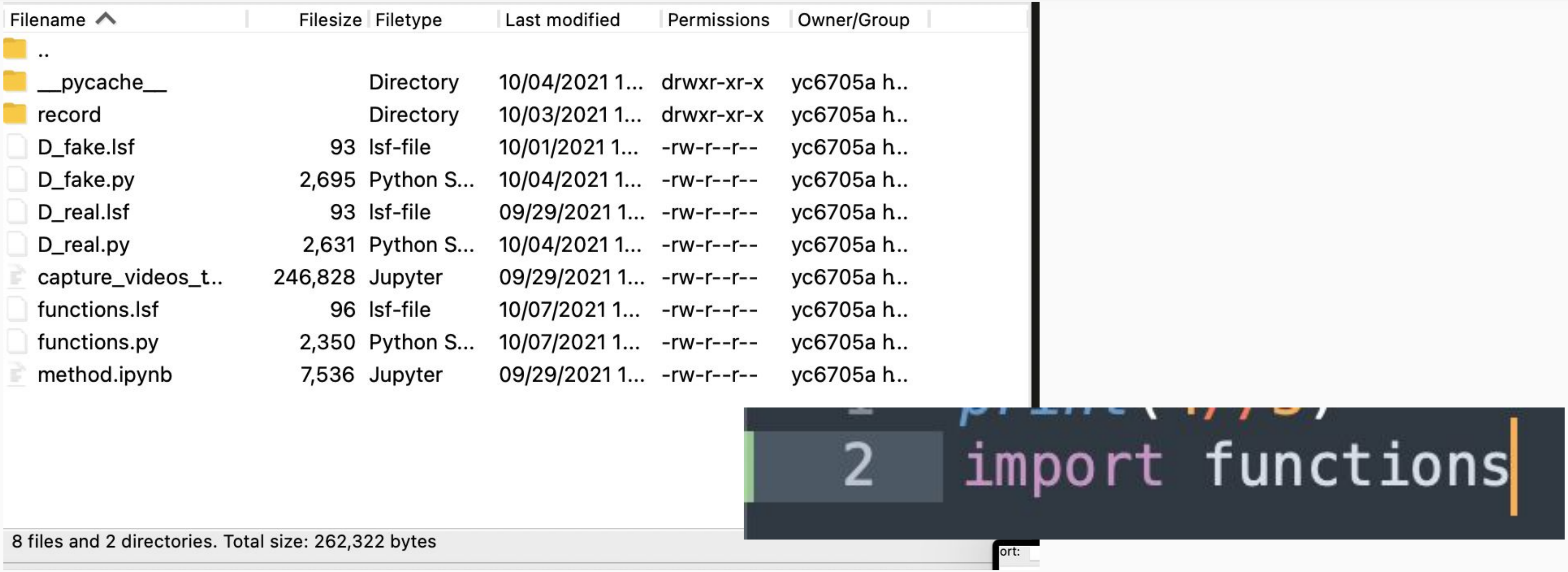
WK6 Report

Replicate the paper

Yunting Chiu

Collect all functions into a func.py

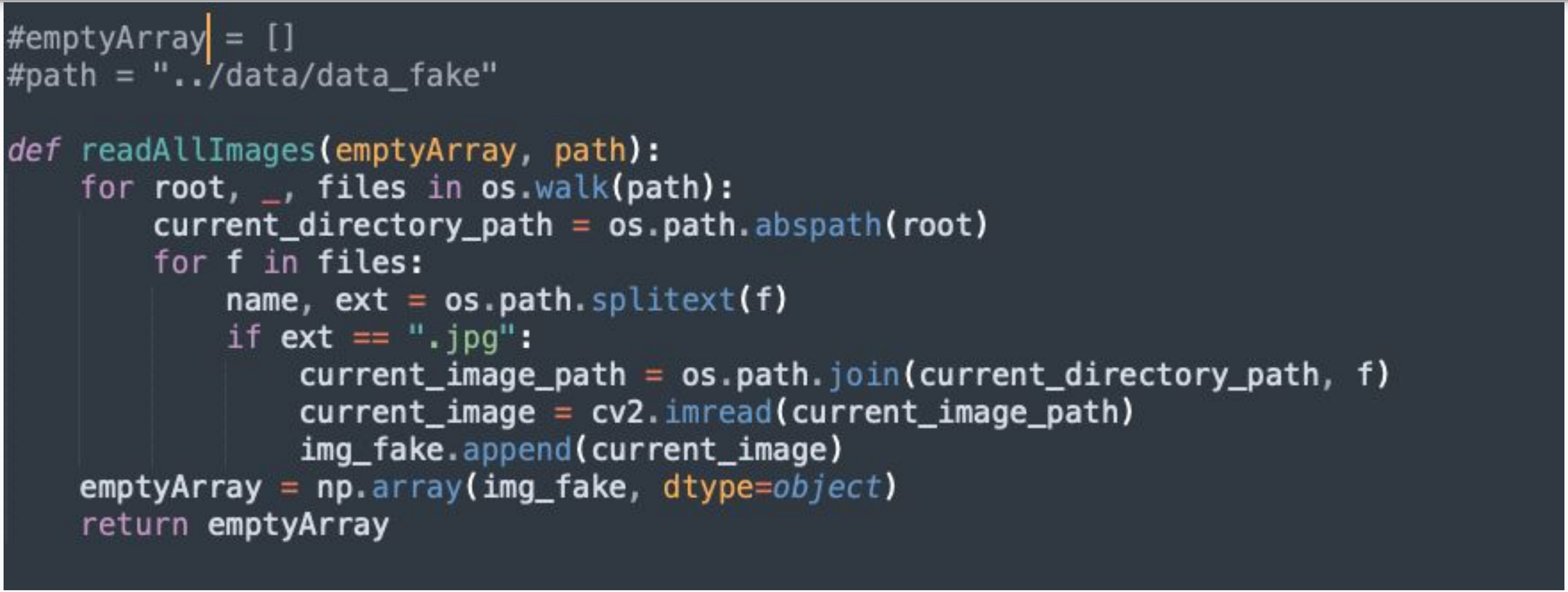


Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group
..					
__pycache__		Directory	10/04/2021 1...	drwxr-xr-x	yc6705a h..
record		Directory	10/03/2021 1...	drwxr-xr-x	yc6705a h..
D_fake.lsf	93	lsf-file	10/01/2021 1...	-rw-r--r--	yc6705a h..
D_fake.py	2,695	Python S...	10/04/2021 1...	-rw-r--r--	yc6705a h..
D_real.lsf	93	lsf-file	09/29/2021 1...	-rw-r--r--	yc6705a h..
D_real.py	2,631	Python S...	10/04/2021 1...	-rw-r--r--	yc6705a h..
capture_videos_t..	246,828	Jupyter	09/29/2021 1...	-rw-r--r--	yc6705a h..
functions.lsf	96	lsf-file	10/07/2021 1...	-rw-r--r--	yc6705a h..
functions.py	2,350	Python S...	10/07/2021 1...	-rw-r--r--	yc6705a h..
method.ipynb	7,536	Jupyter	09/29/2021 1...	-rw-r--r--	yc6705a h..

8 files and 2 directories. Total size: 262,322 bytes

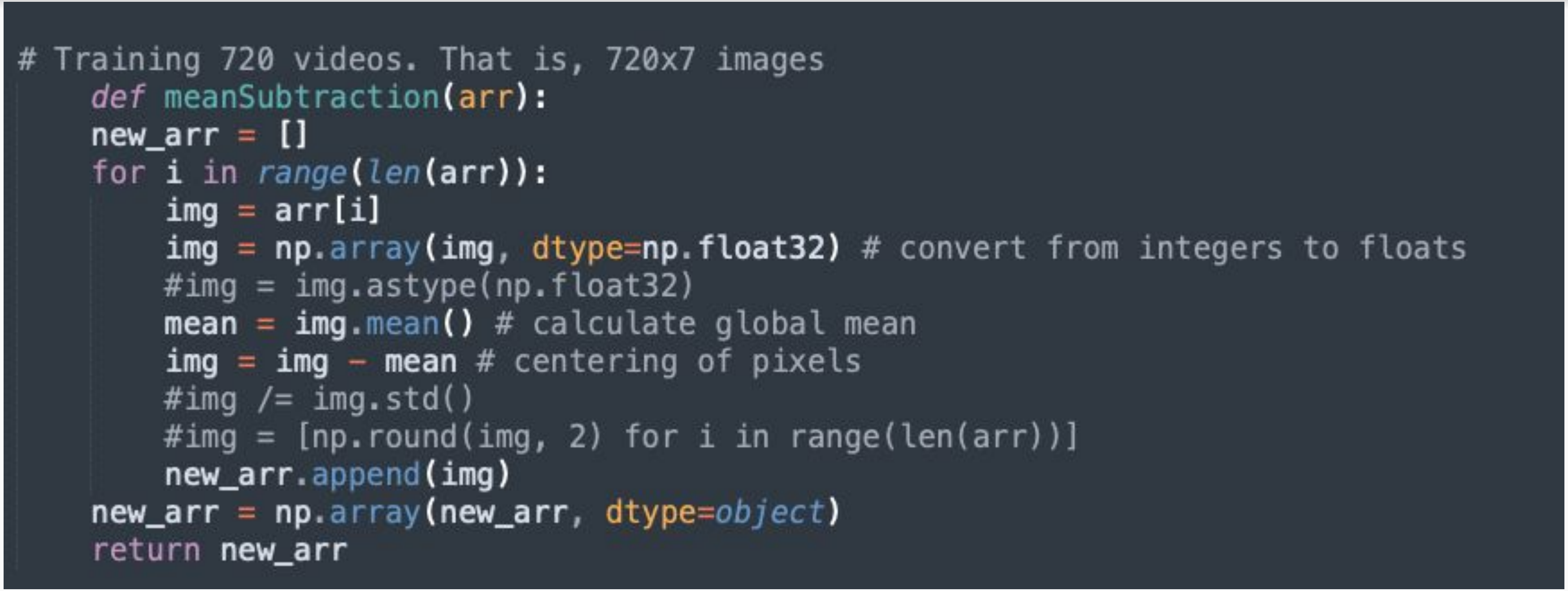
```
import functions
```

Collect all functions into a func.py



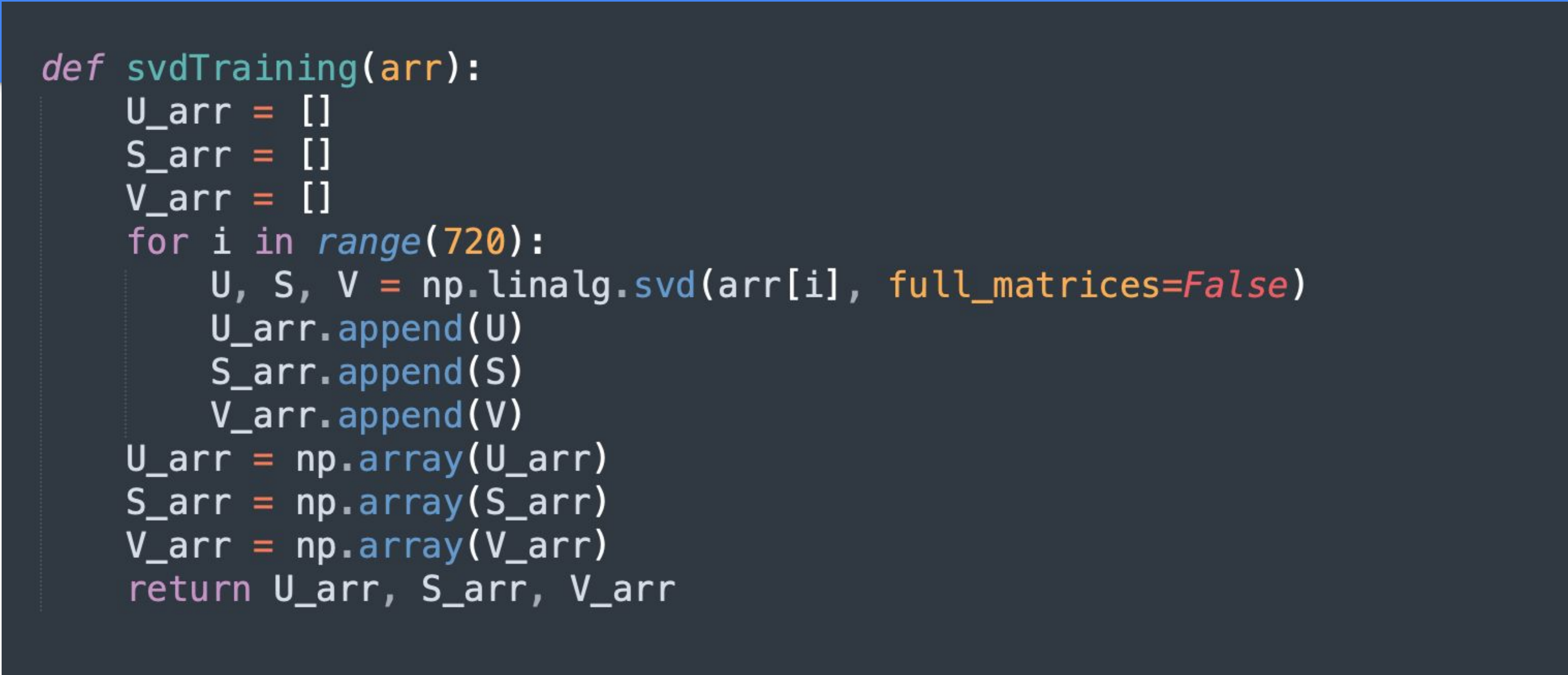
```
#emptyArray = []
#path = "../data/data_fake"

def readAllImages(emptyArray, path):
    for root, _, files in os.walk(path):
        current_directory_path = os.path.abspath(root)
        for f in files:
            name, ext = os.path.splitext(f)
            if ext == ".jpg":
                current_image_path = os.path.join(current_directory_path, f)
                current_image = cv2.imread(current_image_path)
                img_fake.append(current_image)
    emptyArray = np.array(img_fake, dtype=object)
    return emptyArray
```

Collect all functions into a func.py



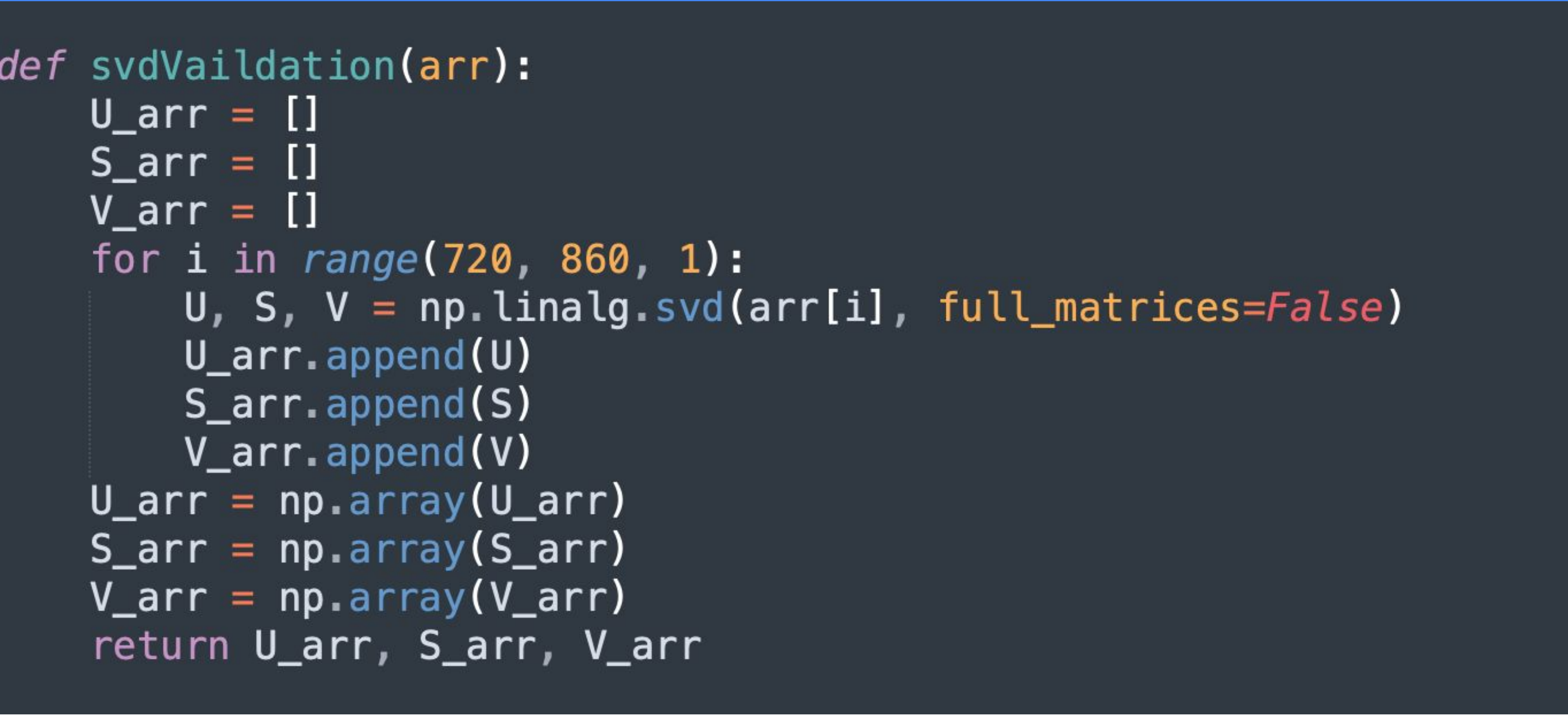
```
# Training 720 videos. That is, 720x7 images
    def meanSubtraction(arr):
    new_arr = []
    for i in range(len(arr)):
        img = arr[i]
        img = np.array(img, dtype=np.float32) # convert from integers to floats
        #img = img.astype(np.float32)
        mean = img.mean() # calculate global mean
        img = img - mean # centering of pixels
        #img /= img.std()
        #img = [np.round(img, 2) for i in range(len(arr))]
        new_arr.append(img)
    new_arr = np.array(new_arr, dtype=object)
    return new_arr
```

Collect all functions into a func.py



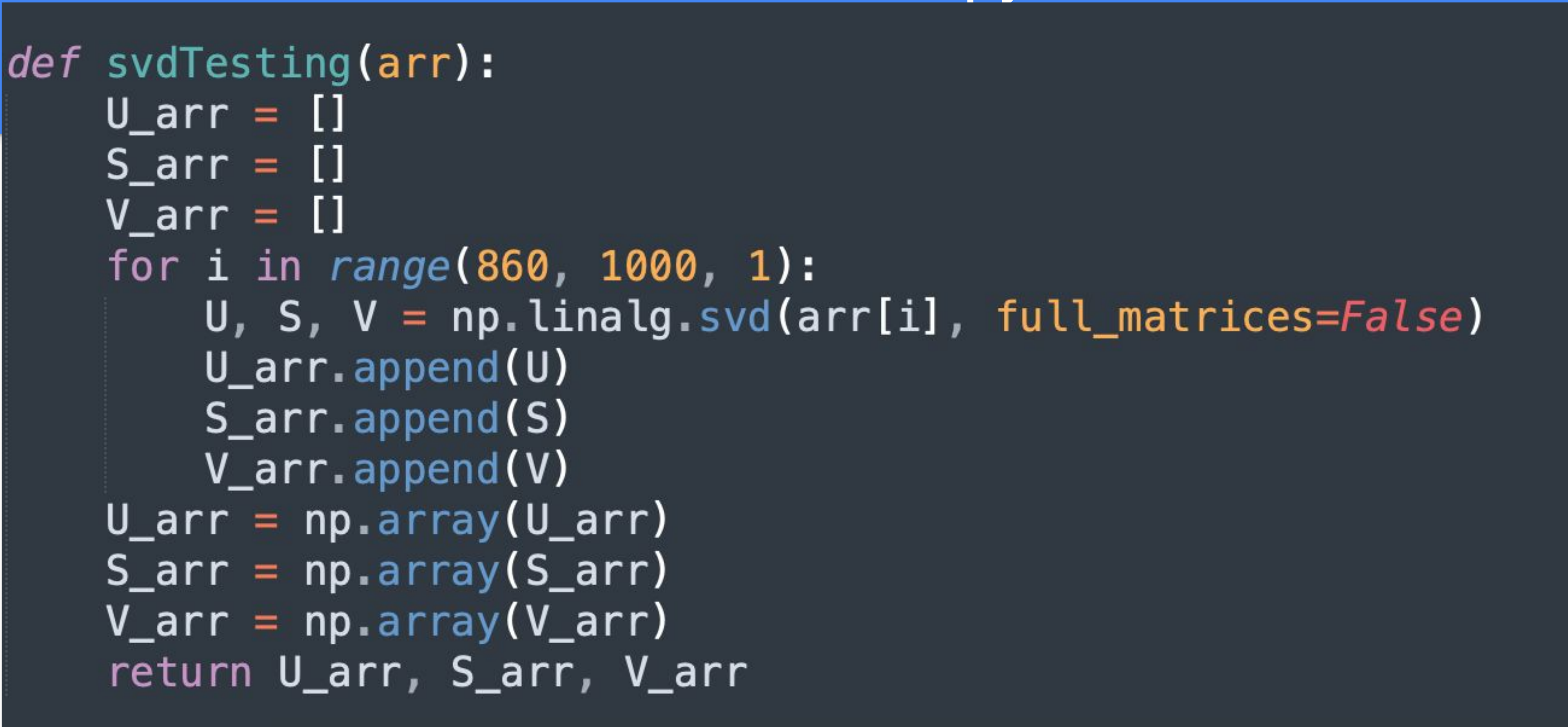
```python
def svdTraining(arr):
    U_arr = []
    S_arr = []
    V_arr = []
    for i in range(720):
        U, S, V = np.linalg.svd(arr[i], full_matrices=False)
        U_arr.append(U)
        S_arr.append(S)
        V_arr.append(V)
    U_arr = np.array(U_arr)
    S_arr = np.array(S_arr)
    V_arr = np.array(V_arr)
    return U_arr, S_arr, V_arr
```

Collect all functions into a func.py



```python
def svdVaildation(arr):
    U_arr = []
    S_arr = []
    V_arr = []
    for i in range(720, 860, 1):
        U, S, V = np.linalg.svd(arr[i], full_matrices=False)
        U_arr.append(U)
        S_arr.append(S)
        V_arr.append(V)
    U_arr = np.array(U_arr)
    S_arr = np.array(S_arr)
    V_arr = np.array(V_arr)
    return U_arr, S_arr, V_arr
```

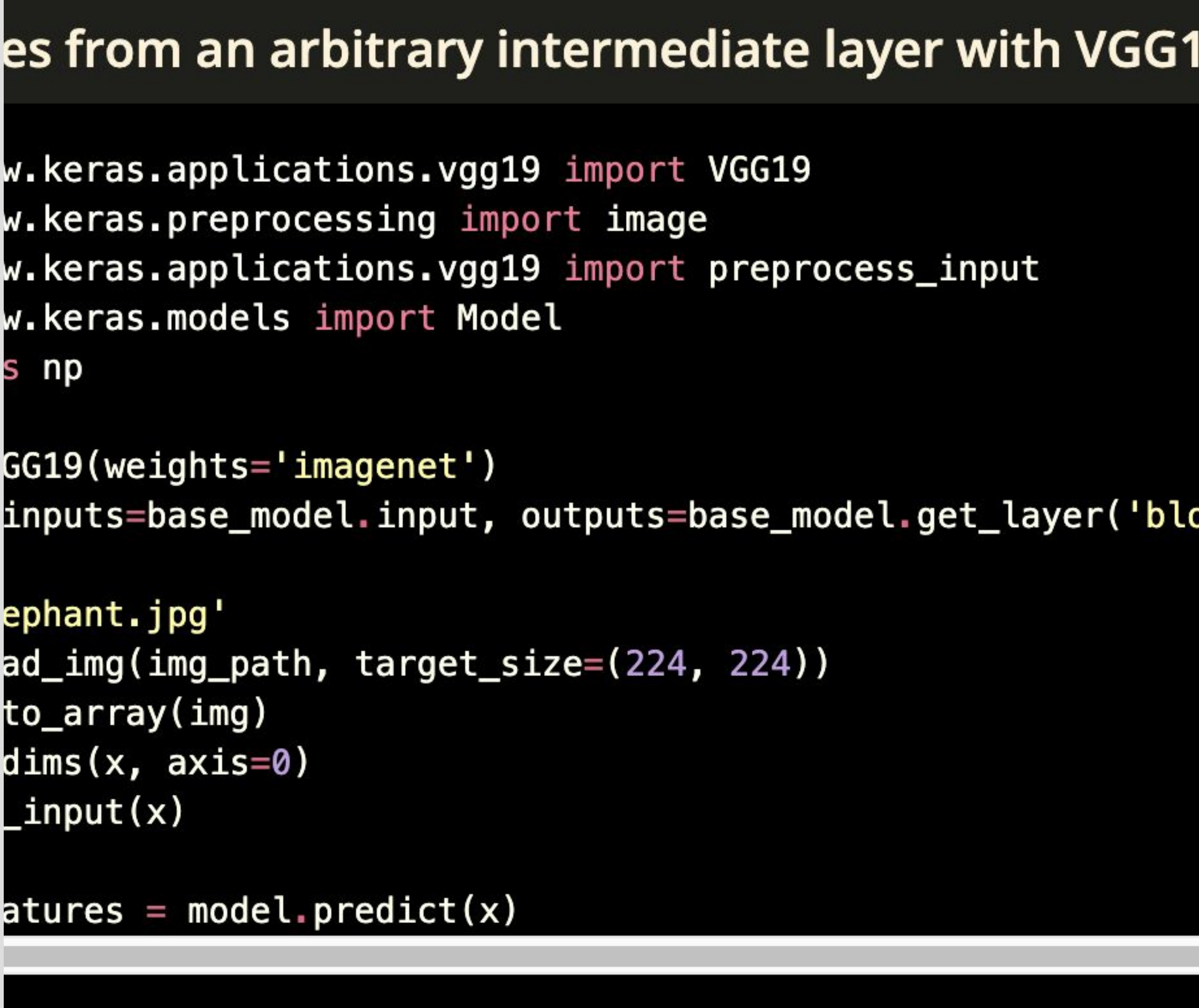
Collect all functions into a func.py



```python
def svdTesting(arr):
    U_arr = []
    S_arr = []
    V_arr = []
    for i in range(860, 1000, 1):
        U, S, V = np.linalg.svd(arr[i], full_matrices=False)
        U_arr.append(U)
        S_arr.append(S)
        V_arr.append(V)
    U_arr = np.array(U_arr)
    S_arr = np.array(S_arr)
    V_arr = np.array(V_arr)
    return U_arr, S_arr, V_arr
```

Keras Applications

- Keras Applications are deep learning models that are made available alongside pre-trained weights
- These models can be used for prediction, feature extraction, and fine-tuning

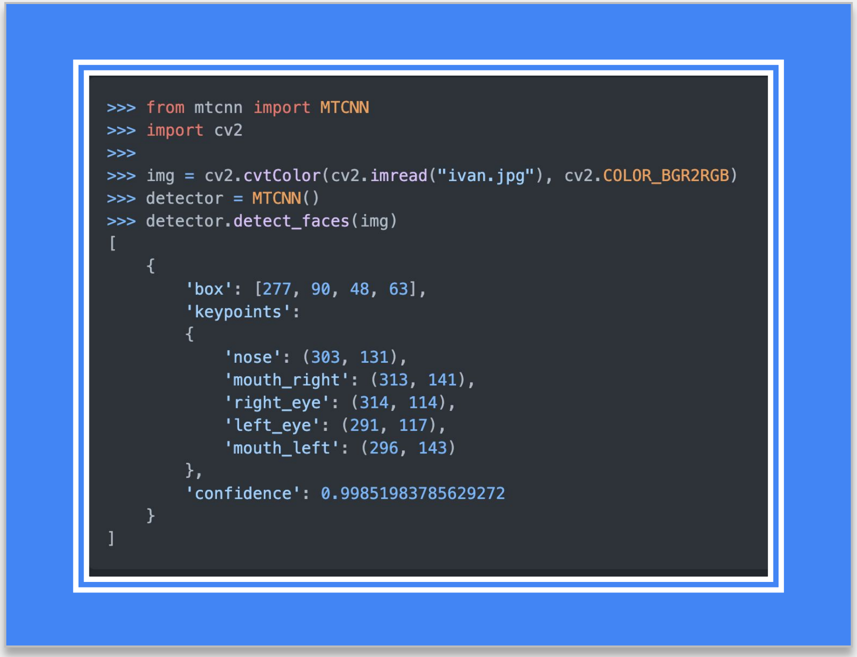


es from an arbitrary intermediate layer with VGG1

```
w.keras.applications.vgg19 import VGG19
w.keras.preprocessing import image
w.keras.applications.vgg19 import preprocess_input
w.keras.models import Model
s np

GG19(weights='imagenet')
inputs=base_model.input, outputs=base_model.get_layer('blo

ephant.jpg'
ad_img(img_path, target_size=(224, 224))
to_array(img)
dims(x, axis=0)
_input(x)

atures = model.predict(x)
```

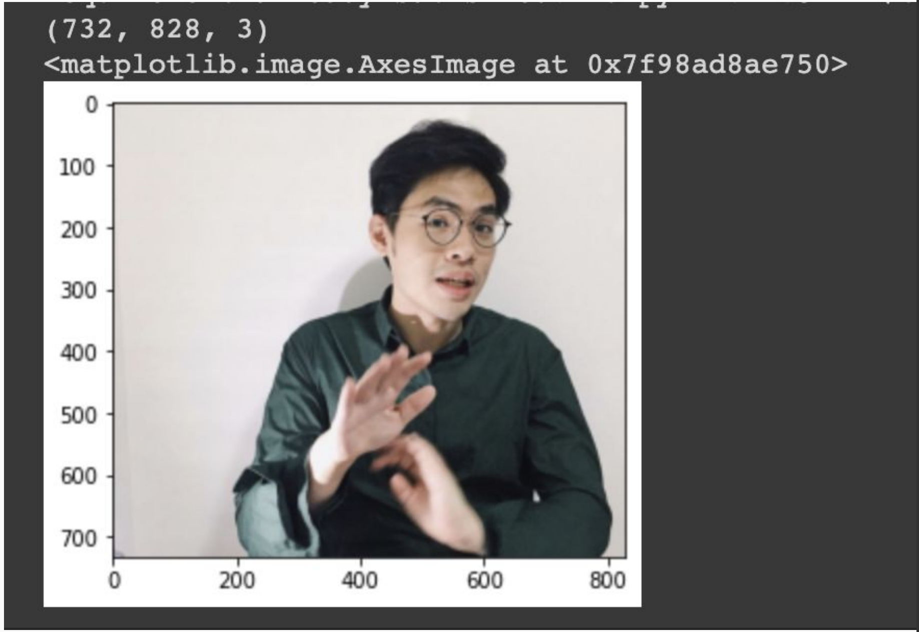
MTCNN face detector

- Box: [x, y, width, height], xy is a pixel position, wh is a bounding
- Conf: probability for a bounding box to be matching a face
- keypoints: pixel position (x, y)

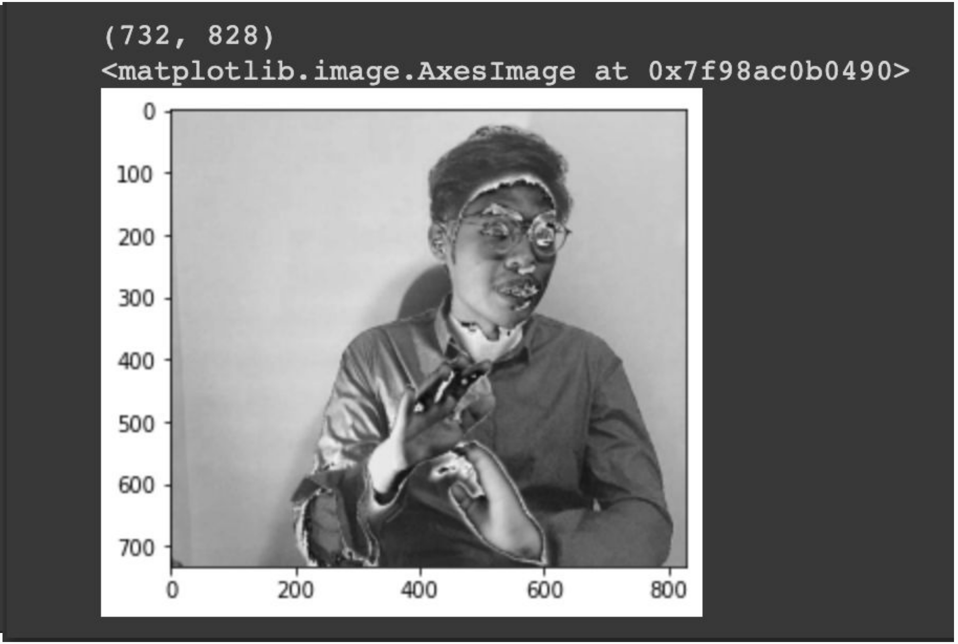


```
>>> from mtcnn import MTCNN
>>> import cv2
>>>
>>> img = cv2.cvtColor(cv2.imread("ivan.jpg"), cv2.COLOR_BGR2RGB)
>>> detector = MTCNN()
>>> detector.detect_faces(img)
[
    {
        'box': [277, 90, 48, 63],
        'keypoints':
        {
            'nose': (303, 131),
            'mouth_right': (313, 141),
            'right_eye': (314, 114),
            'left_eye': (291, 117),
            'mouth_left': (296, 143)
        },
        'confidence': 0.99851983785629272
    }
]
```

MTCNN Example

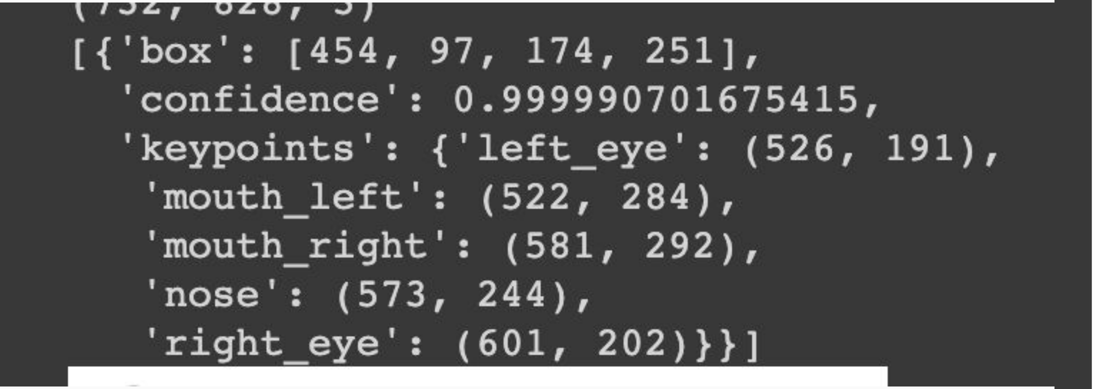


```
(732, 828, 3)
<matplotlib.image.AxesImage at 0x7f98ad8ae750>
```

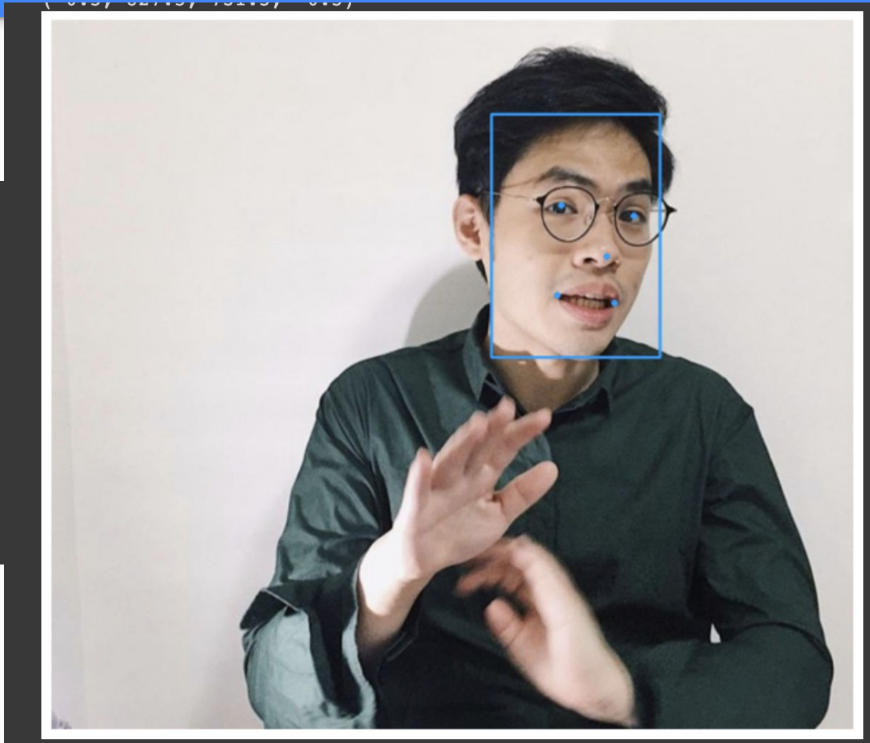


```
(732, 828)
<matplotlib.image.AxesImage at 0x7f98ac0b0490>
```

MTCNN Example



```
(732, 828, 3)
[{'box': [454, 97, 174, 251],
  'confidence': 0.9999907016754 15,
  'keypoints': {'left_eye': (526, 191),
   'mouth_left': (522, 284),
   'mouth_right': (581, 292),
   'nose': (573, 244),
   'right_eye': (601, 202)}}]
```



This Week

1. The authors segment the outer facial ring to identify whether the video is fake or real.
2. Me and Dr. Boukouvalas will use another technique to construct facial features
3. Read the paper, practice the relevant pre-trained model
4. Enjoy the lovely weather