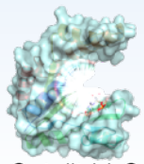
DataWknds.

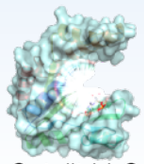
CASE STUDIES



Catalit LLC

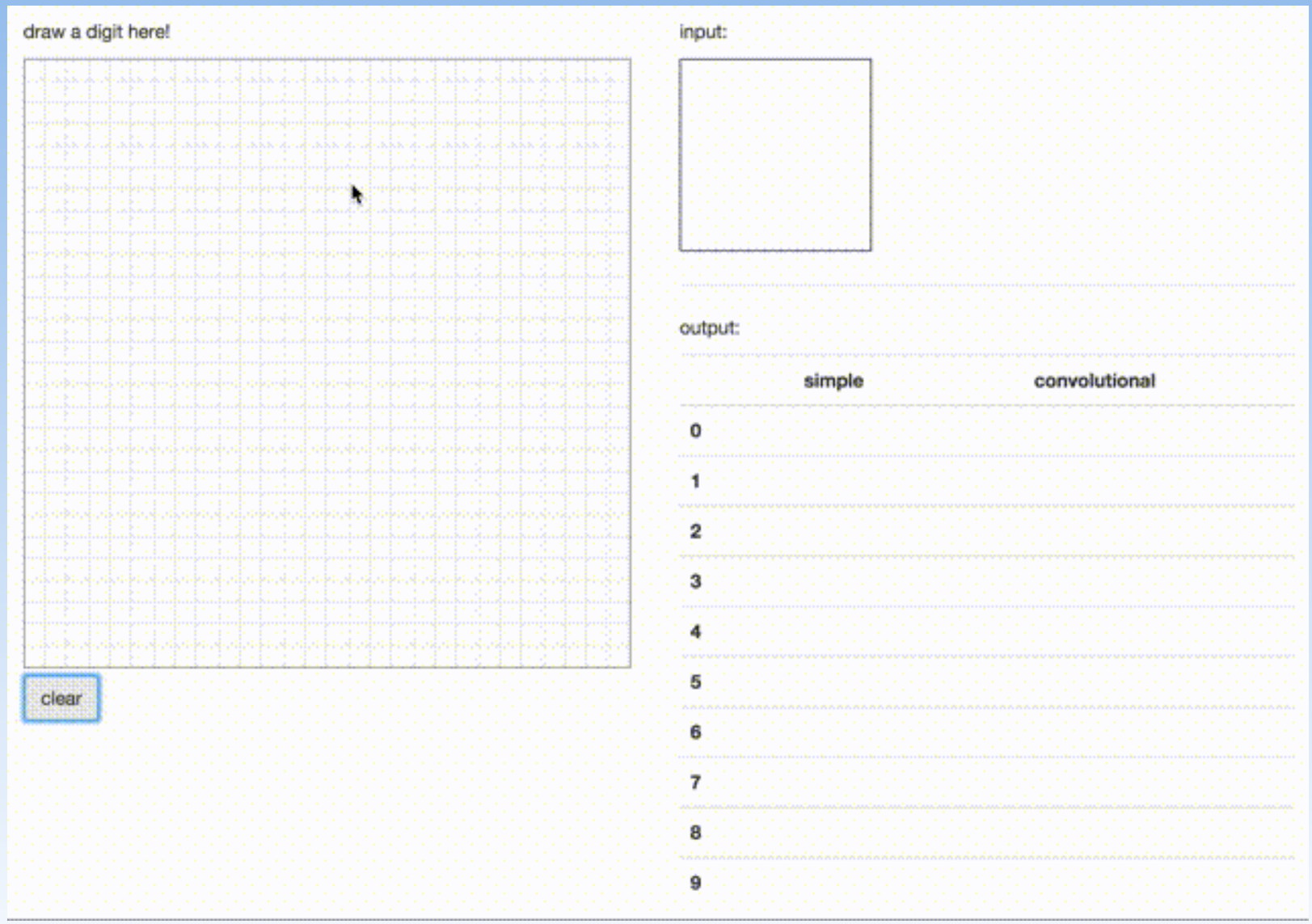
CASE STUDY

TENSORFLOW-MNIST



Catalit LLC

TENSORFLOW-MNIST

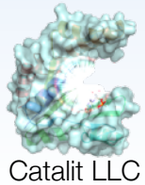


Catalit LLC

```
x = tf.placeholder("float", [None, 784])
sess = tf.Session()

# restore trained data
with tf.variable_scope("regression"):
    y1, variables = model.regression(x)
saver = tf.train.Saver(variables)
saver.restore(sess, "mnist/data/regression.ckpt")


with tf.variable_scope("convolutional"):
    keep_prob = tf.placeholder("float")
    y2, variables = model.convolutional(x, keep_prob)
saver = tf.train.Saver(variables)
saver.restore(sess, "mnist/data/convolutional.ckpt")


def regression(input):
    return sess.run(y1, feed_dict={x: input}).flatten().tolist()


def convolutional(input):
    return sess.run(y2, feed_dict={x: input, keep_prob: 1.0}).flatten().tolist()
```

CODE

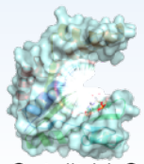
```
33  # webapp
34  app = Flask(__name__)
35
36
37  @app.route('/api/mnist', methods=['POST'])
38  def mnist():
39      input = ((255 - np.array(request.json, dtype=np.uint8)) / 255.0).reshape(1, 784)
40      output1 = regression(input)
41      output2 = convolutional(input)
42      return jsonify(results=[output1, output2])
43
44
45  @app.route('/')
46  def main():
47      return render_template('index.html')
48
49
50  if __name__ == '__main__':
51      app.run()
```



Catalit LLC

SUMMARY

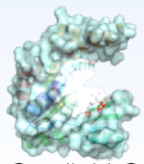
- WHY IS IT IMPORTANT?
- model save / restore
- flask serving of model prediction



Catalit LLC

RUN IT!

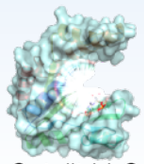
- source activate dwdlagpu
- cd 6_case_studies/tensorflow-mnist
- ./run.sh
- point browser to: http://<your-ip>:8000



Catalit LLC

CASE STUDY

NEURAL STYLE TRANSFER

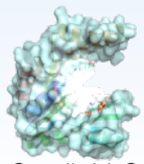


Catalit LLC

NEURAL STYLE TRANSFER

SUMMARY

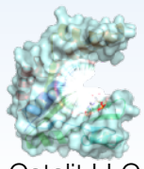
- WHY IS IT IMPORTANT?
- custom loss
- uses pre-trained VGG model



Catalit LLC

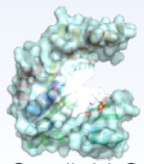
RUN IT!

- source activate dwdlagpu
- cd 6_case_studies/neural_style_transfer
- python neural_style_transfer.py img/obama.jpg img/starry_night.jpg results/obamagogh
- to download:
  - from laptop do:
  - scp -r -i dataweekends.pem ubuntu@<your-ip>:~/DL_Advanced_Workshop-03-11-2017/6_case_studies/neural_style_transfer/results .



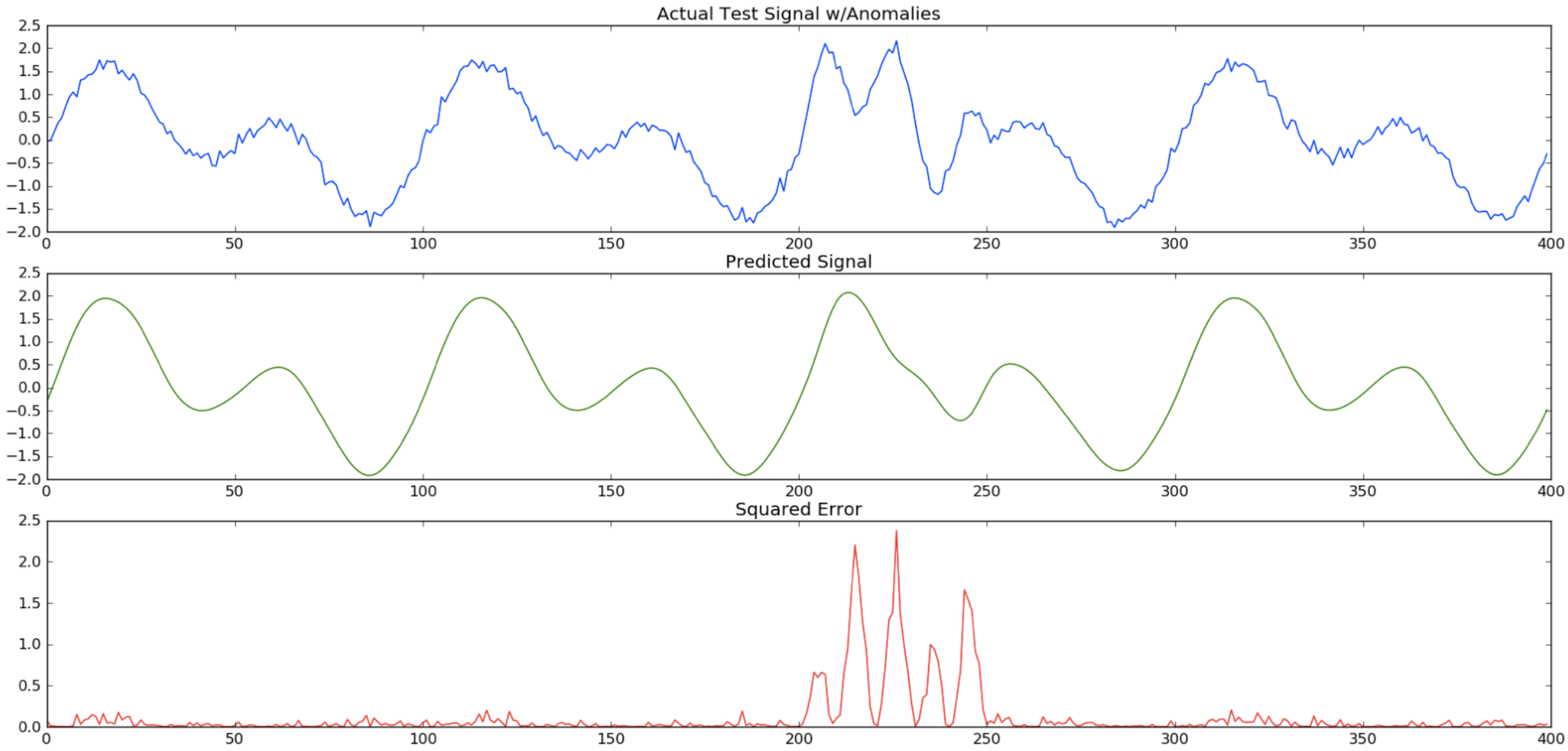
CASE STUDY

LSTM ANOMALY DETECTION



Catalit LLC

LSTM-ANOMALY



https://github.com/ghego/lstm-anomaly-detect

Catalit LLC

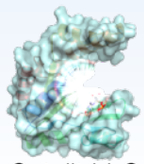
CODE

```
115  def build_model():
116      model = Sequential()
117      layers = {'input': 1, 'hidden1': 64, 'hidden2': 256, 'hidden3': 100, 'output': 1}
118
119      model.add(LSTM(
120              input_length=sequence_length - 1,
121              input_dim=layers['input'],
122              output_dim=layers['hidden1'],
123              return_sequences=True))
124      model.add(Dropout(0.2))
125
126      model.add(LSTM(
127              layers['hidden2'],
128              return_sequences=True))
129      model.add(Dropout(0.2))
130
131      model.add(LSTM(
132              layers['hidden3'],
133              return_sequences=False))
134      model.add(Dropout(0.2))
135
136      model.add(Dense(
137              output_dim=layers['output']))
138      model.add(Activation("linear"))
139
140      start = time.time()
141      model.compile(loss="mse", optimizer="rmsprop")
142      print "Compilation Time : ", time.time() - start
143      return model
```

Catalit LLC

SUMMARY

- WHY IS IT IMPORTANT?
- LSTM on time series
- stacked LSTM

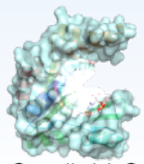


Catalit LLC

RUN IT!

- cd 6_case_studies/lstm-anomaly-detection

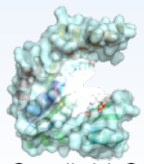
- python lstm-synthetic-wave-anomaly-detect.py



Catalit LLC

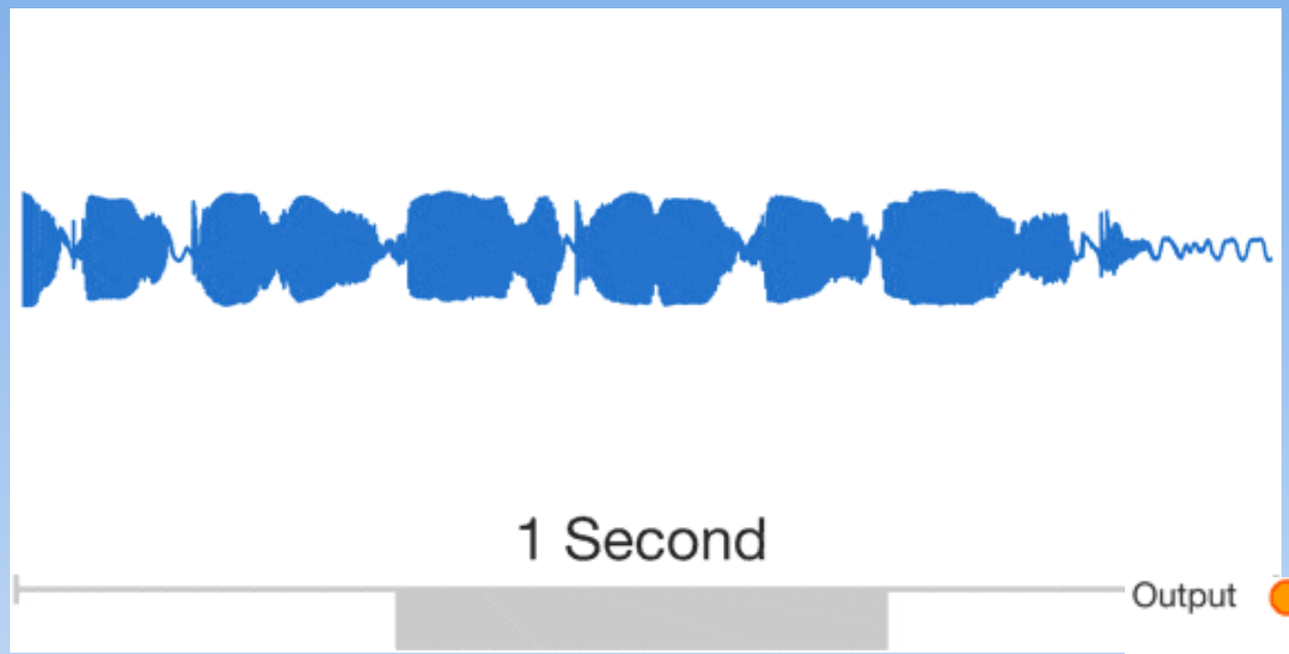
CASE STUDY

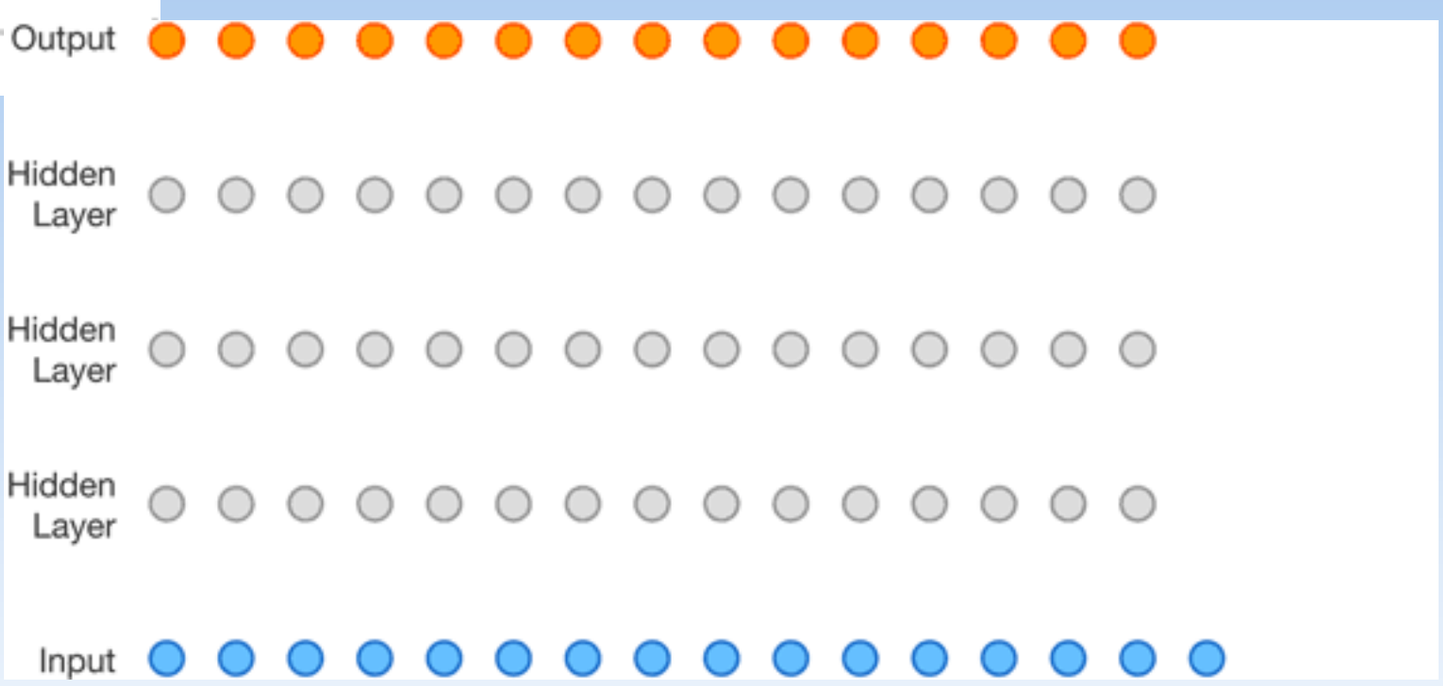
WAVENET



Catalit LLC

WAVENET

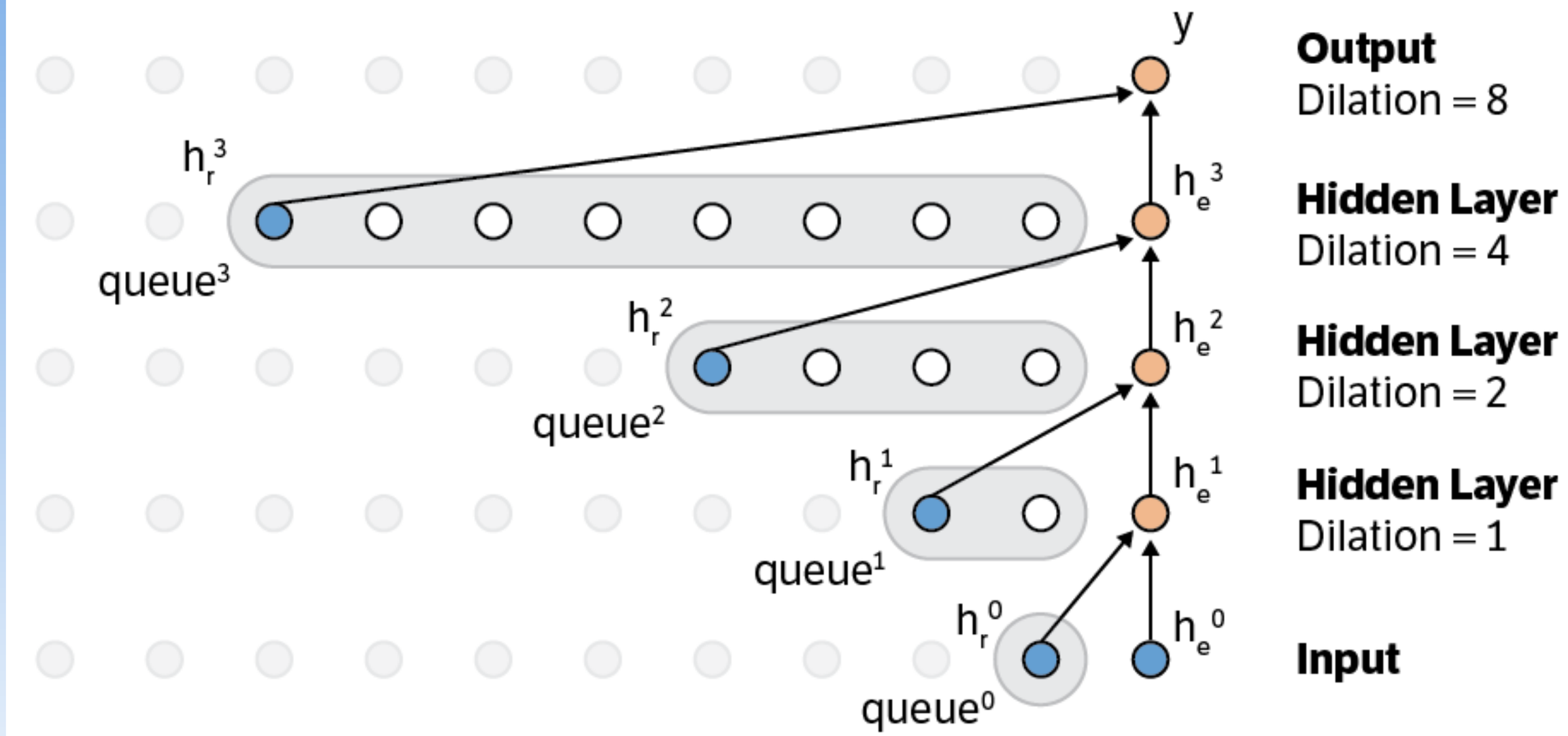




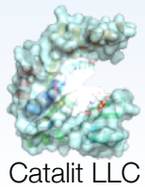
https://deepmind.com/blog/wavenet-generative-model-raw-audio/

Catalit LLC

FAST-WAVENET

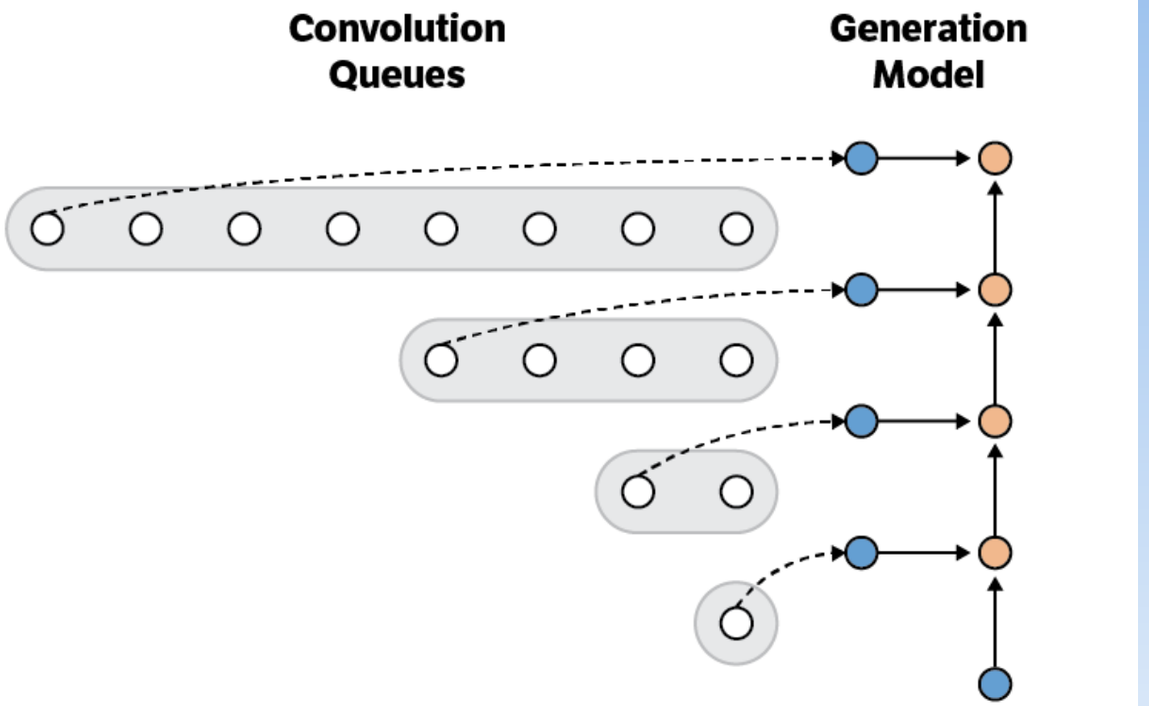


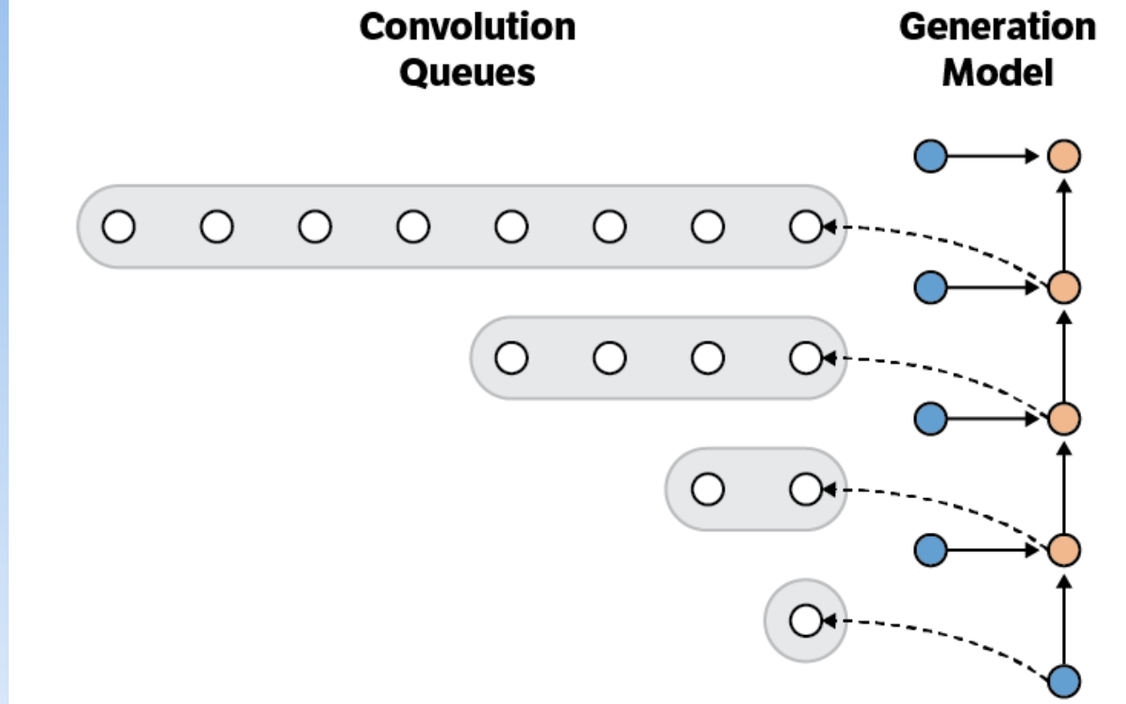
https://github.com/tomlepaine/fast-wavenet

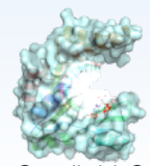


Catalit LLC

IMPLEMENTATION



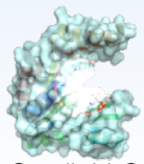




Catalit LLC

SUMMARY

- WHY IS IT IMPORTANT?
- stacked convolution
- convolutions with gaps
- pixel level model

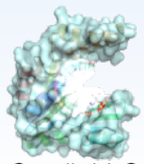


Catalit LLC

RUN IT!

- open jupyter notebook

- browse to folder 6_case_studies/fast-wavenet

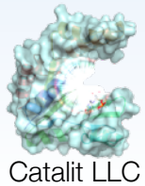


Catalit LLC

SEE ALSO

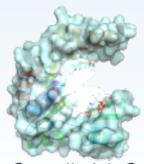
```
def wavenetBlock(n_atrous_filters, atrous_filter_size, atrous_rate):
    def f(input_):
        residual = input_
        tanh_out = AtrousConvolution1D(n_atrous_filters, atrous_filter_size,
                                       atrous_rate=atrous_rate,
                                       border_mode='same',
                                       activation='tanh')(input_)
        sigmoid_out = AtrousConvolution1D(n_atrous_filters, atrous_filter_size,
                                          atrous_rate=atrous_rate,
                                          border_mode='same',
                                          activation='sigmoid')(input_)
        merged = merge([tanh_out, sigmoid_out], mode='mul')
        skip_out = Convolution1D(1, 1, activation='relu', border_mode='same')(merged)
        out = merge([skip_out, residual], mode='sum')
        return out, skip_out
    return f
```

https://github.com/usernaamee/keras-wavenet



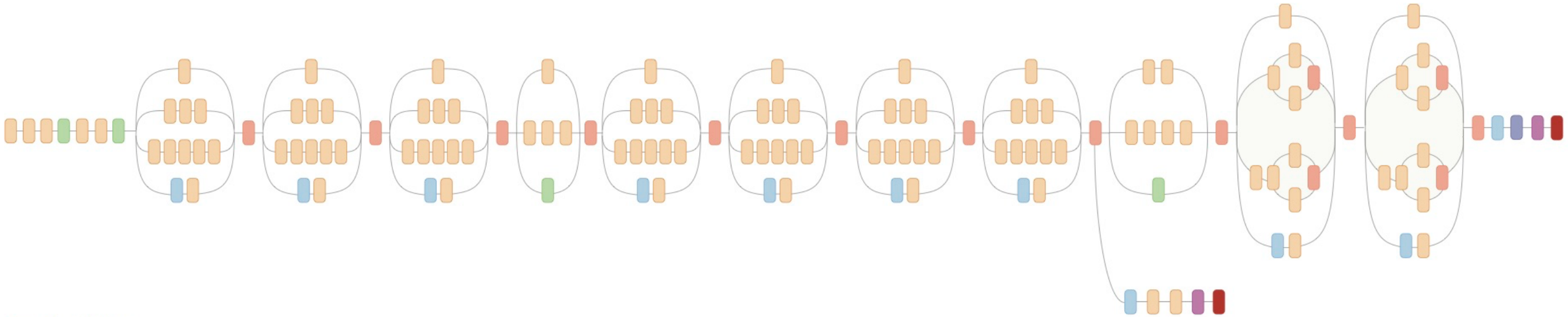
CASE STUDY

INCEPTION RETRAIN



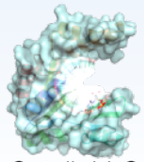
Catalit LLC

INCEPTION



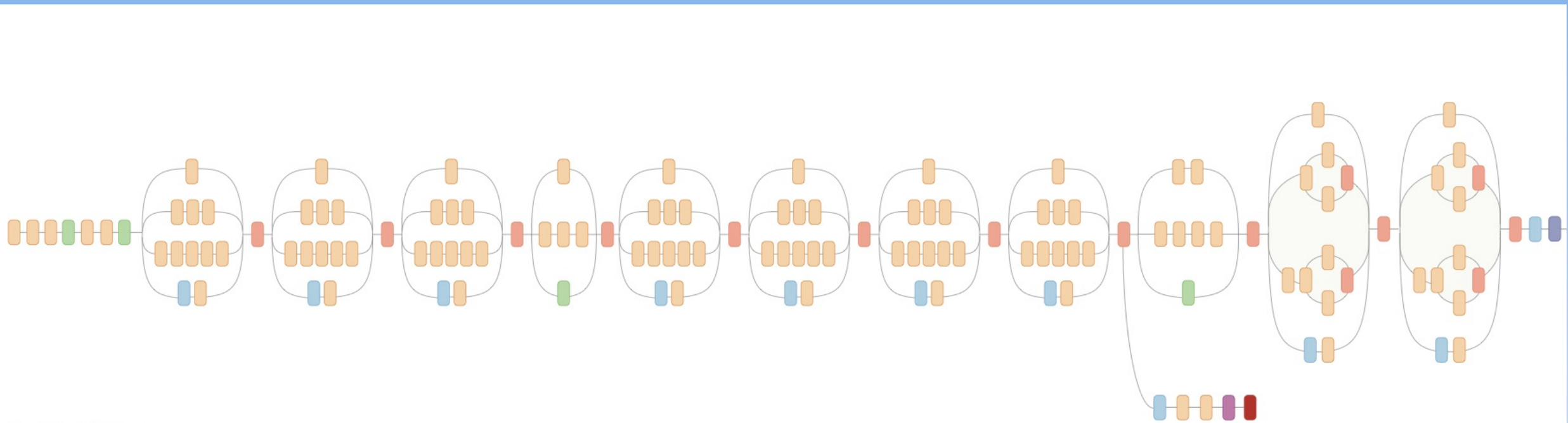
- Convolution
- AvgPool
- MaxPool
- Concat
- Dropout
- Fully connected
- Softmax

http://arxiv.org/abs/1512.00567



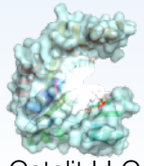
Catalit LLC

INCEPTION



Convolution
AvgPool
MaxPool
Concat
Dropout
Fully connected
Softmax

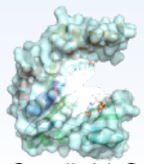
REUSE AS IS



Catalit LLC

INCEPTION

RETRAIN :)



Catalit LLC

DETECT FLOWERS

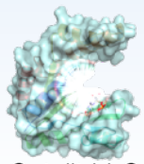
Catalit LLC

RUN IT!

- cd 6_case_studies/inception-retrain

- ./retrain.sh

- python ./label_image.py ./tf_files/flower_photos/daisy/21652746_cc379e0eea_m.jpg



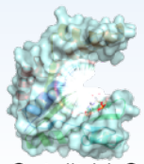
Catalit LLC

CODE

```
DATA_URL = 'http://download.tensorflow.org/models/image/imagenet/inception-2015-12-05.tgz'
# pylint: enable=line-too-long
BOTTLENECK_TENSOR_NAME = 'pool_3/_reshape:0'
BOTTLENECK_TENSOR_SIZE = 2048
```

```
JPEG_DATA_TENSOR_NAME = 'DecodeJpeg/contents:0'
RESIZED_INPUT_TENSOR_NAME = 'ResizeBilinear:0'
MAX_NUM_IMAGES_PER_CLASS = 2 ** 27 - 1  # ~134M
```

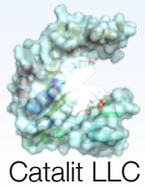
```
def create_inception_graph():
  """Creates a graph from saved GraphDef file and returns a Graph object.

  Returns:
    Graph holding the trained Inception network, and various tensors we'll be
    manipulating.
  """
  with tf.Session() as sess:
    model_filename = os.path.join(
        FLAGS.model_dir, 'classify_image_graph_def.pb')
    with gfile.FastGFile(model_filename, 'rb') as f:
      graph_def = tf.GraphDef()
      graph_def.ParseFromString(f.read())
      bottleneck_tensor, jpeg_data_tensor, resized_input_tensor = (
          tf.import_graph_def(graph_def, name='', return_elements=[
              BOTTLENECK_TENSOR_NAME, JPEG_DATA_TENSOR_NAME,
              RESIZED_INPUT_TENSOR_NAME]))
  return sess.graph, bottleneck_tensor, jpeg_data_tensor, resized_input_tensor
```



Catalit LLC

BOTTLENECKS

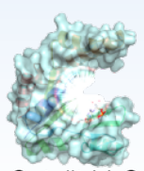
```
def run_bottleneck_on_image(sess, image_data, image_data_tensor,
                            bottleneck_tensor):
  """Runs inference on an image to extract the 'bottleneck' summary layer.

  Args:
    sess: Current active TensorFlow Session.
    image_data: String of raw JPEG data.
    image_data_tensor: Input data layer in the graph.
    bottleneck_tensor: Layer before the final softmax.

  Returns:
    Numpy array of bottleneck values.
  """
  bottleneck_values = sess.run(
      bottleneck_tensor,
      {image_data_tensor: image_data})
  bottleneck_values = np.squeeze(bottleneck_values)
  return bottleneck_values
```



Catalit LLC

RETRAINING

```
# Organizing the following ops as `final_training_ops` so they're easier
# to see in TensorBoard
layer_name = 'final_training_ops'
with tf.name_scope(layer_name):
    with tf.name_scope('weights'):
        layer_weights = tf.Variable(tf.truncated_normal(
            [BOTTLENECK_TENSOR_SIZE, class_count], stddev=0.001), name='final_weights')
        variable_summaries(layer_weights)
    with tf.name_scope('biases'):
        layer_biases = tf.Variable(
            tf.zeros([class_count]), name='final_biases')
        variable_summaries(layer_biases)
    with tf.name_scope('Wx_plus_b'):
        logits = tf.matmul(bottleneck_input, layer_weights) + layer_biases
        tf.summary.histogram('pre_activations', logits)
```



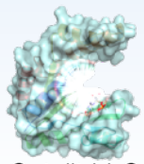
Catalit LLC

INCEPTION SUMMARY

- WHY IS IT IMPORTANT?
  - shows how to re-use a pre-trained network
  - piggyback on shoulders of giants
  - tensorflow name scopes
  - data augmentation
  - argument parsing

CASE STUDY

DEEP ANPR



Catalit LLC

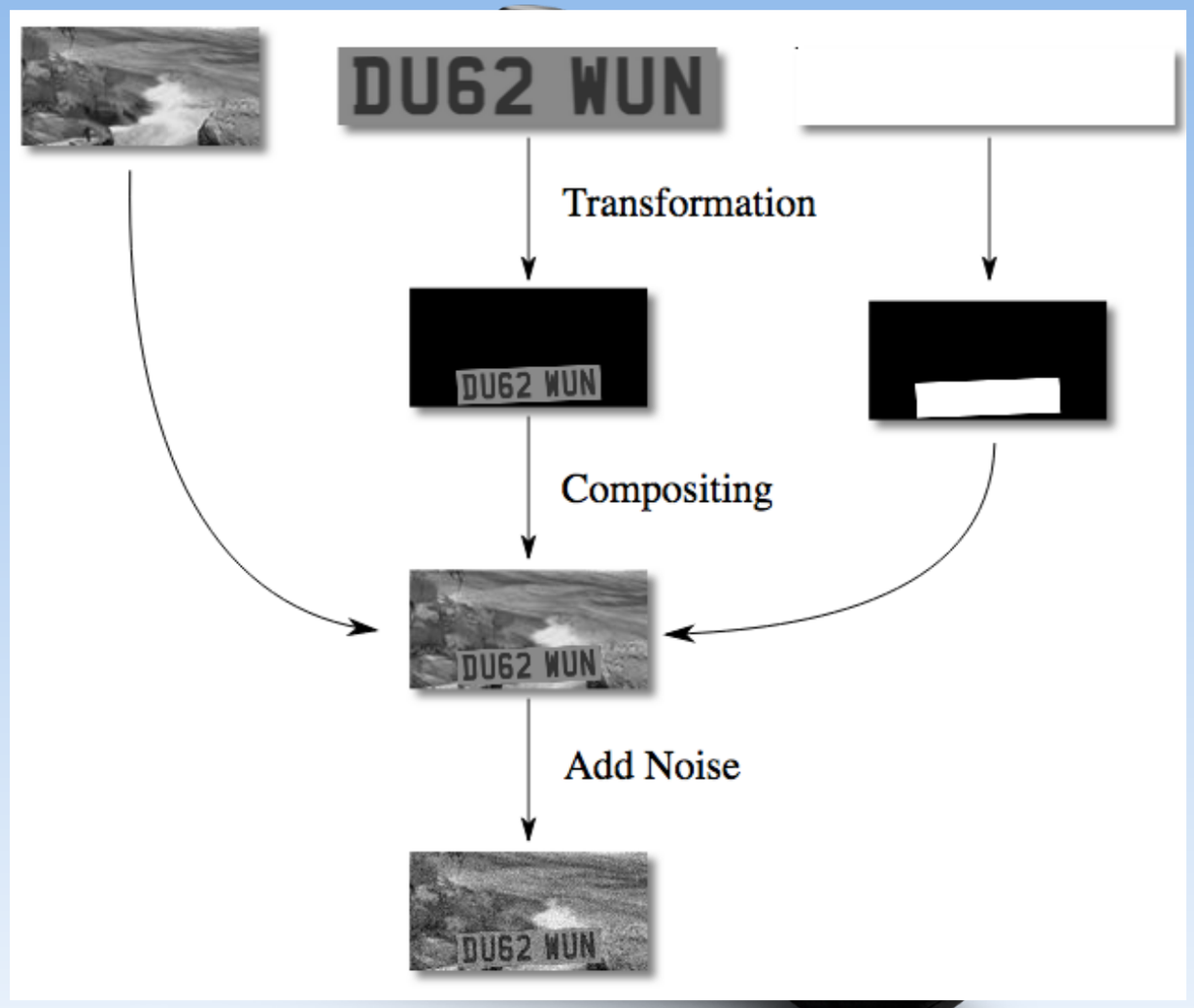
PLATE RECOGNITION WITH CNN

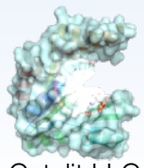
http://matthewearl.github.io/2016/05/06/cnn-anpr/

IMAGE SYNTHESIS

- synthetic images generated in the cpu during training

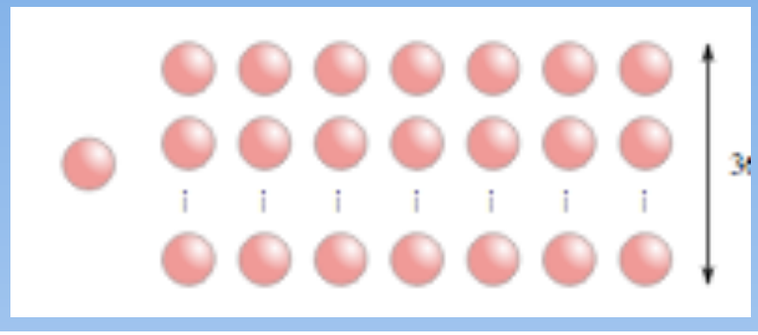
- using opencv





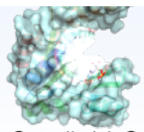
LOSS FUNCTION

- 1 bit for presence
  => sigmoid

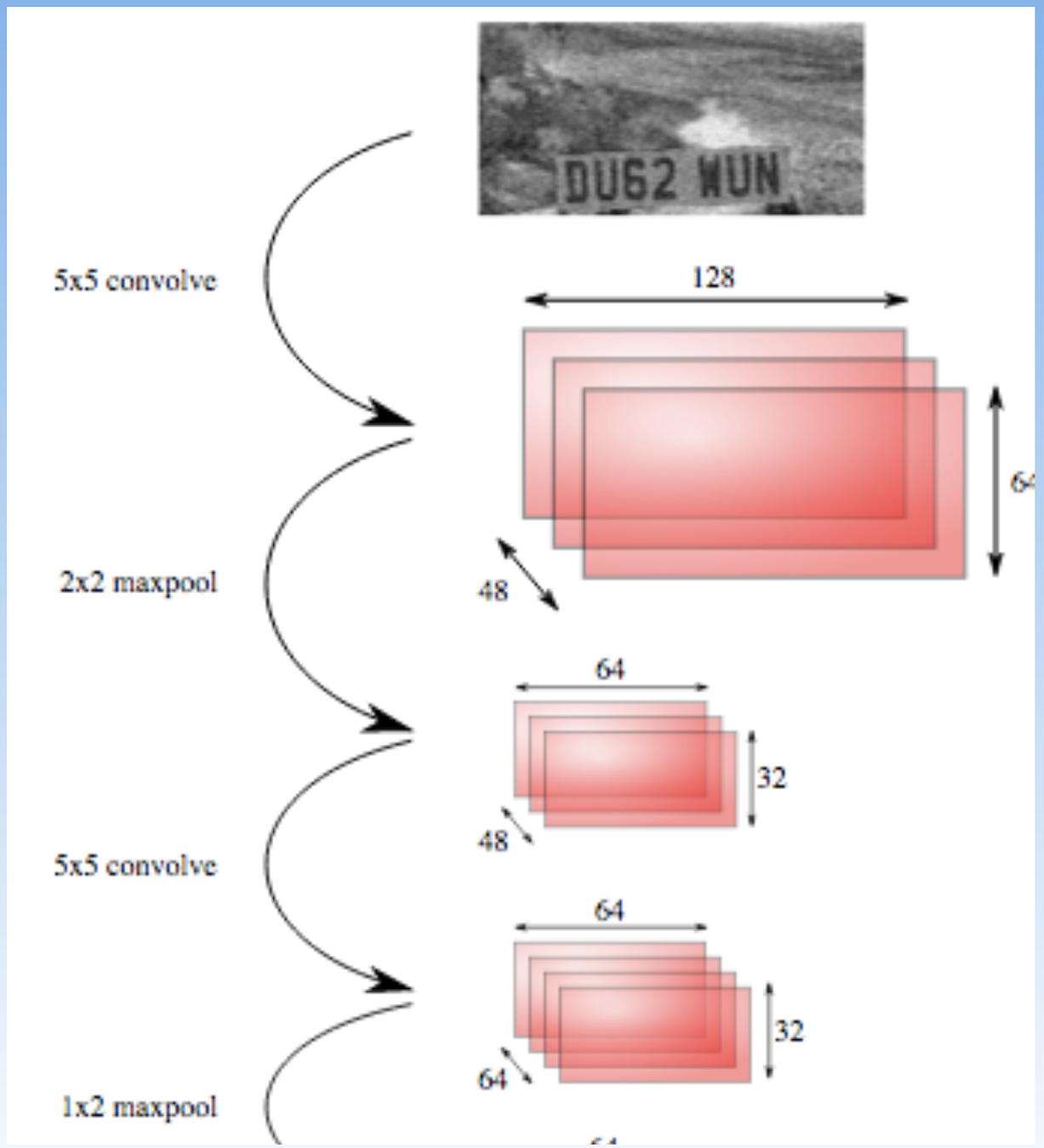


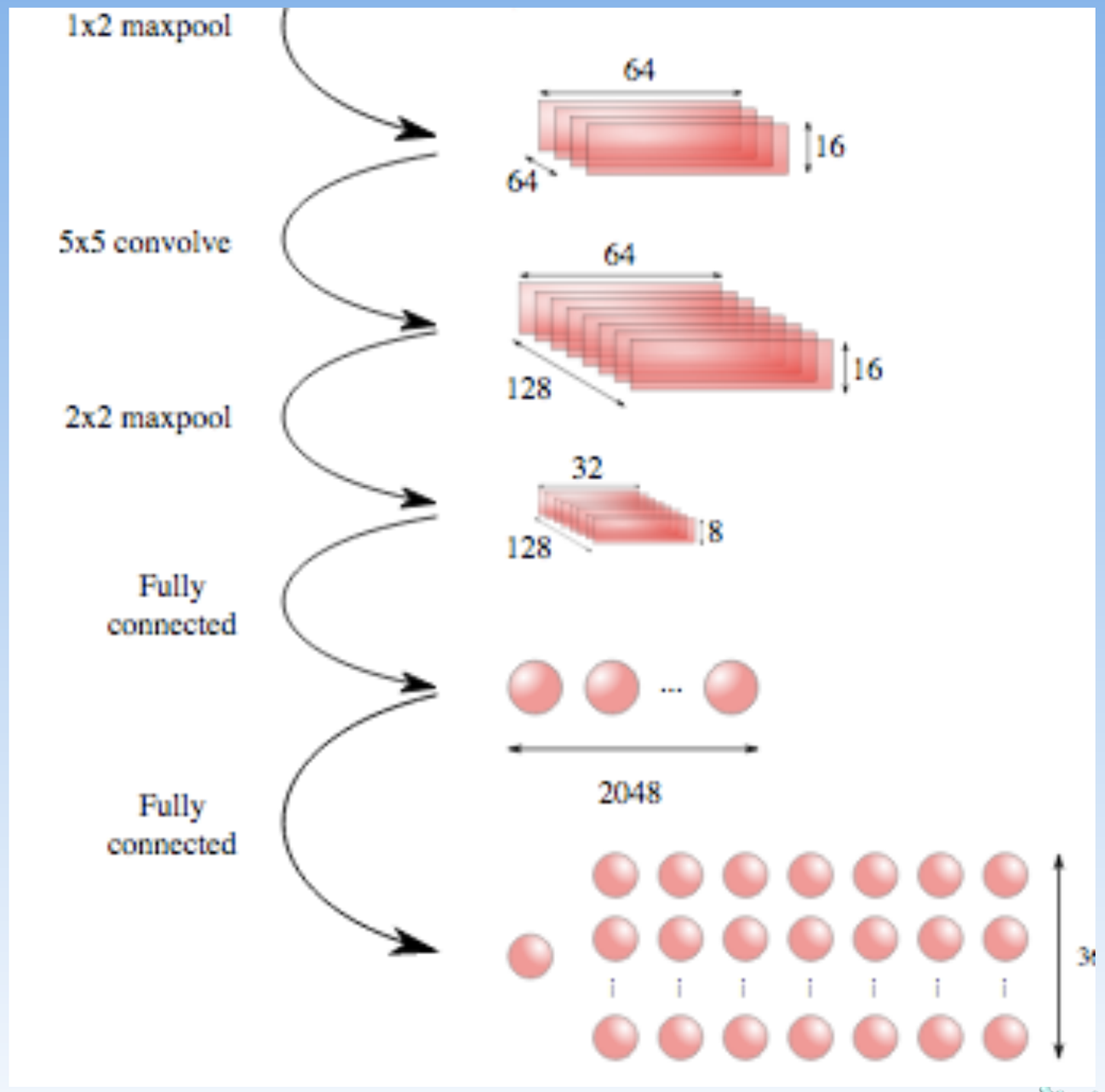
- 7 chars plate (out of 36)
  => softmax on each

- expected output `HH41RFP 1`.
- expected output `FB78PFD 1`.
- expected output `JW01GAI 0`. (Plate partially truncated.)
- expected output `AM46KVG 0`. (Plate too small.)
- expected output `XG86KIO 0`. (Plate too big.)
- expected output `XH07NYO 0`. (Plate not present at all.)



MODEL





Catalit LLC

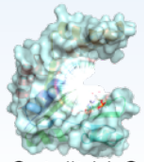
PREDICTION

- get prediction at every window

- clever trick => map the fully connected part to convolutional layers

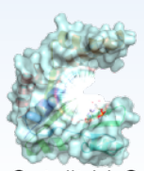
RUN IT!

- cd 6_case_studies/deep-anpr
- ./train.py
- when tired stop it
- ./detect.py test/<some_image>.png weights.npz out.png



CODE

```
def generate_im(char_ims, num_bg_images):
    bg = generate_bg(num_bg_images)

    plate, plate_mask, code = generate_plate(FONT_HEIGHT, char_ims)

    M, out_of_bounds = make_affine_transform(
                            from_shape=plate.shape,
                            to_shape=bg.shape,
                            min_scale=0.6,
                            max_scale=0.875,
                            rotation_variation=1.0,
                            scale_variation=1.5,
                            translation_variation=1.2)
    plate = cv2.warpAffine(plate, M, (bg.shape[1], bg.shape[0]))
    plate_mask = cv2.warpAffine(plate_mask, M, (bg.shape[1], bg.shape[0]))

    out = plate * plate_mask + bg * (1.0 - plate_mask)

    out = cv2.resize(out, (OUTPUT_SHAPE[1], OUTPUT_SHAPE[0]))

    out += numpy.random.normal(scale=0.05, size=out.shape)
    out = numpy.clip(out, 0., 1.)

    return out, code, not out_of_bounds
```



Catalit LLC

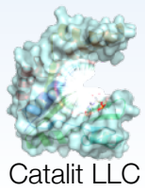
CODE

```python
def get_loss(y, y_):
    # Calculate the loss from digits being incorrect.  Don't count loss from
    # digits that are in non-present plates.
    digits_loss = tf.nn.softmax_cross_entropy_with_logits(
                                            logits=tf.reshape(y[:, 1:],
                                              [-1, len(common.CHARS)]),
                                            labels=tf.reshape(y_[:, 1:],
                                                              [-1, len(common.CHARS)]))
    digits_loss = tf.reshape(digits_loss, [-1, 7])
    digits_loss = tf.reduce_sum(digits_loss, 1)
    digits_loss *= (y_[:, 0] != 0)
    digits_loss = tf.reduce_sum(digits_loss)

    # Calculate the loss from presence indicator being wrong.
    presence_loss = tf.nn.sigmoid_cross_entropy_with_logits(
        logits=y[:, :1], labels=y_[:, :1])
    presence_loss = 7 * tf.reduce_sum(presence_loss)

    return digits_loss, presence_loss, digits_loss + presence_loss
```

Catalit LLC

CODE

```python
def convolutional_layers():
    """
    Get the convolutional layers of the model.

    """
    x = tf.placeholder(tf.float32, [None, None, None])

    # First layer
    W_conv1 = weight_variable([5, 5, 1, 48])
    b_conv1 = bias_variable([48])
    x_expanded = tf.expand_dims(x, 3)
    h_conv1 = tf.nn.relu(conv2d(x_expanded, W_conv1) + b_conv1)
    h_pool1 = max_pool(h_conv1, ksize=(2, 2), stride=(2, 2))

    # Second layer
    W_conv2 = weight_variable([5, 5, 48, 64])
    b_conv2 = bias_variable([64])

    h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
    h_pool2 = max_pool(h_conv2, ksize=(2, 1), stride=(2, 1))

    # Third layer
    W_conv3 = weight_variable([5, 5, 64, 128])
    b_conv3 = bias_variable([128])

    h_conv3 = tf.nn.relu(conv2d(h_pool2, W_conv3) + b_conv3)
    h_pool3 = max_pool(h_conv3, ksize=(2, 2), stride=(2, 2))

    return x, h_pool3, [W_conv1, b_conv1,
                        W_conv2, b_conv2,
                        W_conv3, b_conv3]
```

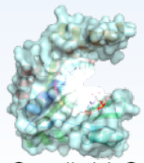


Catalit LLC

CODE

```
get_training_model():
"""
The training model acts on a batch of 128x64 windows, and outp
7 * len(common.CHARS) vector, `v`. `v[0]` is the probability t
fully within the image and is at the correct scale.

`v[1 + i * len(common.CHARS) + c]` is the probability that the
character is `c`.

"""
x, conv_layer, conv_vars = convolutional_layers()

# Densely connected layer
W_fc1 = weight_variable([32 * 8 * 128, 2048])
b_fc1 = bias_variable([2048])

conv_layer_flat = tf.reshape(conv_layer, [-1, 32 * 8 * 128])
h_fc1 = tf.nn.relu(tf.matmul(conv_layer_flat, W_fc1) + b_fc1)

# Output layer
W_fc2 = weight_variable([2048, 1 + 7 * len(common.CHARS)])
b_fc2 = bias_variable([1 + 7 * len(common.CHARS)])

y = tf.matmul(h_fc1, W_fc2) + b_fc2

return (x, y, conv_vars + [W_fc1, b_fc1, W_fc2, b_fc2])
```

```
def get_detect_model():
    """
    The same as the training model, except it acts on an arbitrarily size
    input, and slides the 128x64 window across the image in 8x8 strides.

    The output is of the form `v`, where `v[i, j]` is equivalent to the o
    of the training model, for the window at coordinates `(8 * i, 4 * j)`

    """
    x, conv_layer, conv_vars = convolutional_layers()

    # Fourth layer
    W_fc1 = weight_variable([8 * 32 * 128, 2048])
    W_conv1 = tf.reshape(W_fc1, [8,  32, 128, 2048])
    b_fc1 = bias_variable([2048])
    h_conv1 = tf.nn.relu(conv2d(conv_layer, W_conv1,
                                stride=(1, 1), padding="VALID") + b_fc1)
    # Fifth layer
    W_fc2 = weight_variable([2048, 1 + 7 * len(common.CHARS)])
    W_conv2 = tf.reshape(W_fc2, [1, 1, 2048, 1 + 7 * len(common.CHARS)])
    b_fc2 = bias_variable([1 + 7 * len(common.CHARS)])
    h_conv2 = conv2d(h_conv1, W_conv2) + b_fc2

    return (x, h_conv2, conv_vars + [W_fc1, b_fc1, W_fc2, b_fc2])
```

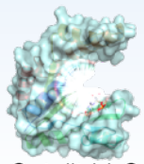


Catalit LLC

DEEP ANPR SUMMARY

- WHY IS IT IMPORTANT?
    - synthetic data augmentation / generation
    - opencv for image preprocessing and generation
    - separate parallel processes in CPU and GPU
    - convolution on arbitrary size image
    - custom loss
    - different way to save model (older code, no checkpoints yet)

LAB



Catalit LLC