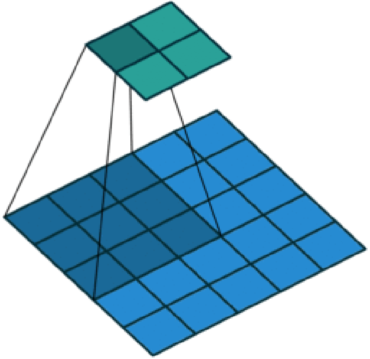
CNN Overview

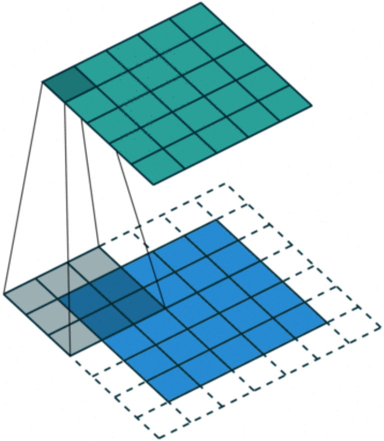
Lukáš Gajdošech

30.08.2020 - 13.09.2020

Convolution I

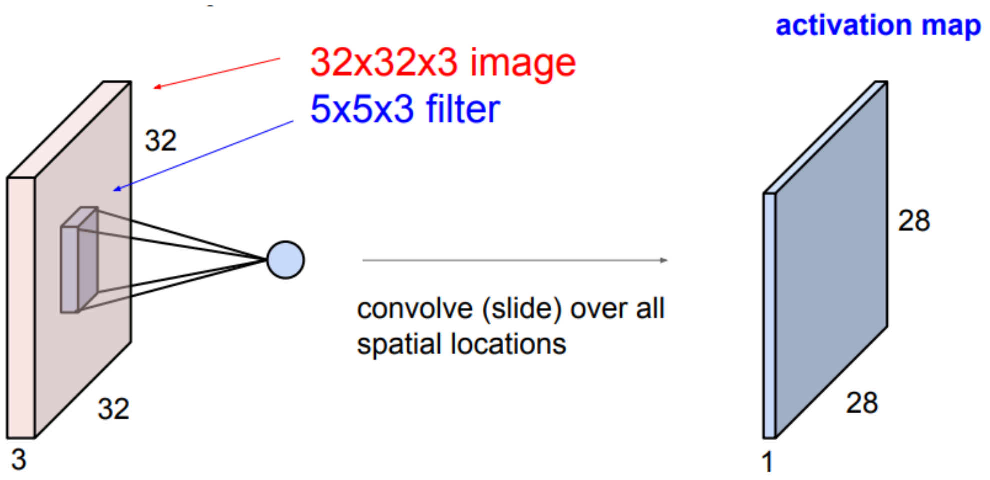


(a) basic



(b) padded

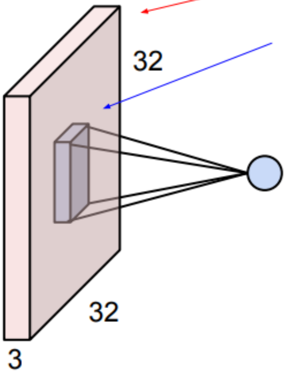
Convolution II



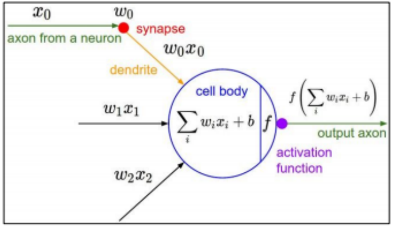
Convolution II

32x32x3 image

5x5x3 filter



1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)



It's just a neuron with local connectivity...

Convolution Math I

$$g(\mathbf{x}; \{K_i\}_{i\in\{1,\ldots,n\}}) = \phi_n(K_n * \phi_{n-1}(K_{n-1} * \cdots * (\phi_1(K_1 * \mathbf{x}))))$$

Figure: CNN Formula, $*$ is the convolution operator, ϕ_i is a non-linearity, K_i for $i \in \{1, \ldots, n\}$ are convolutional kernels and x is the input.

$$K * \mathbf{x} = \begin{pmatrix} k_1 & k_2 \\ k_3 & k_4 \end{pmatrix} * \begin{pmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{pmatrix}$$

Figure: Single $*$ operation.

Convolution Math II

$$K * \mathbf{x} \equiv \begin{pmatrix} k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 & 0 & 0 \\ 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 \\ 0 & 0 & 0 & 0 & k_1 & k_2 & 0 & k_3 & k_4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{pmatrix}$$

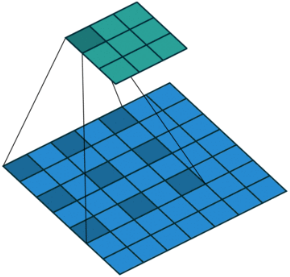
Figure: Convolution as matrix multiplication.

Convolution Math III

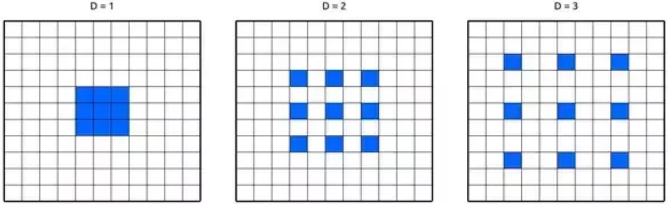
$$g(\mathbf{x}) = u * \mathbf{h} = \begin{pmatrix} u_5 & u_6 & 0 & u_8 & u_9 & 0 & 0 & 0 & 0 \\ u_4 & u_5 & u_6 & u_7 & u_8 & u_9 & 0 & 0 & 0 \\ 0 & u_4 & u_5 & 0 & u_7 & u_8 & 0 & 0 & 0 \\ u_2 & u_3 & 0 & u_5 & u_6 & 0 & u_8 & u_9 & 0 \\ u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 & u_8 & u_9 \\ 0 & u_1 & u_2 & 0 & u_4 & u_5 & 0 & u_7 & u_8 \\ 0 & 0 & 0 & u_2 & u_3 & 0 & u_5 & u_6 & 0 \\ 0 & 0 & 0 & u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \\ 0 & 0 & 0 & 0 & u_1 & u_2 & 0 & u_4 & u_5 \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{pmatrix}$$

Figure: With padded zeros.

Dilated Convolution

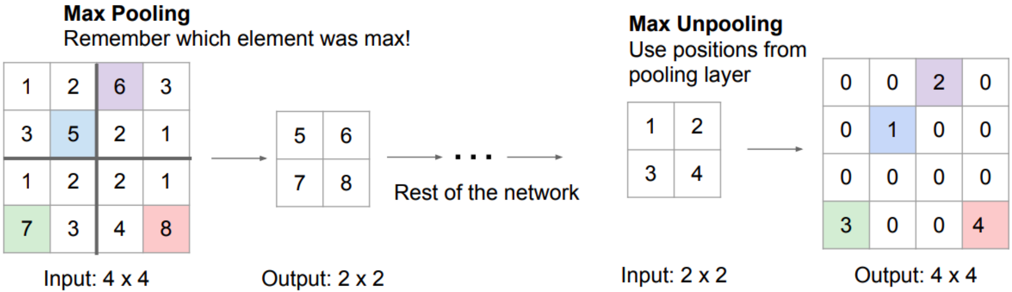


(a) idea



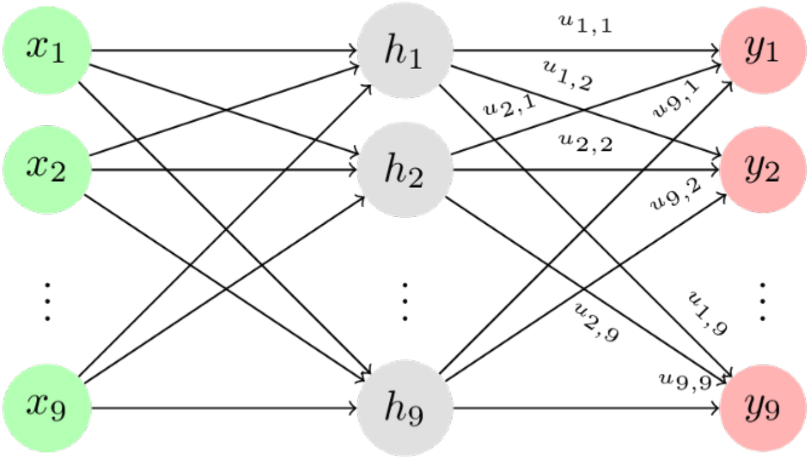
(b) parameter

Pooling and Upsampling

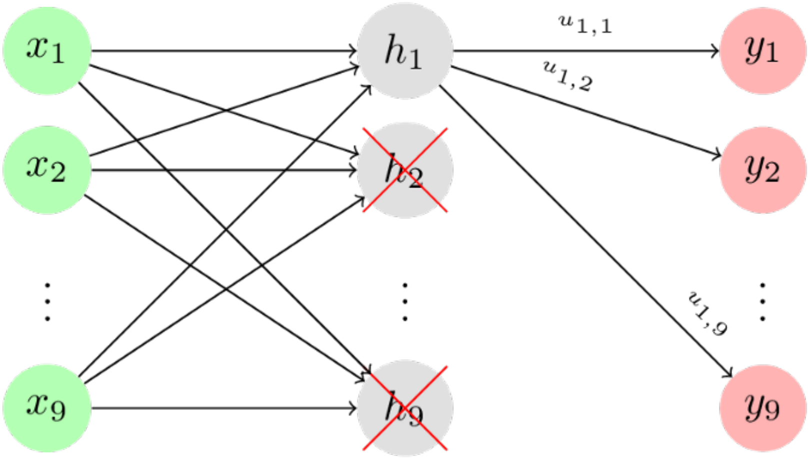


Dropout I

Traditional Neural Network



Dropout II

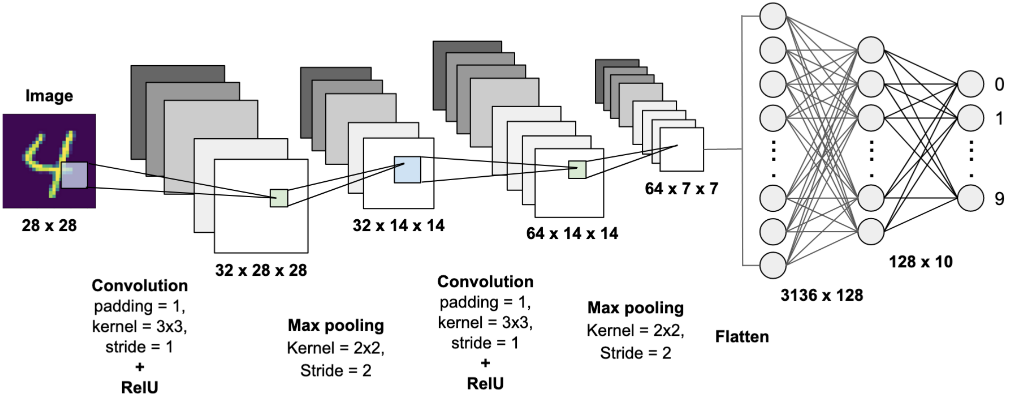


Dropout III

Dropout in CNN, $r_i \in \{0, 1\}$ are random variables.

$$U\,\mathrm{diag}(\mathbf{r})\mathbf{h} = \begin{pmatrix} r_1u_5 & r_2u_6 & 0 & r_4u_8 & r_5u_9 & 0 & 0 & 0 & 0 \\ r_1u_4 & r_2u_5 & r_3u_6 & r_4u_7 & r_5u_8 & r_6u_9 & 0 & 0 & 0 \\ 0 & r_2u_4 & r_3u_5 & 0 & r_5u_7 & r_6u_8 & 0 & 0 & 0 \\ r_1u_2 & r_2u_3 & 0 & r_4u_5 & r_5u_6 & 0 & r_7u_8 & r_8u_9 & 0 \\ r_1u_1 & r_2u_2 & r_3u_3 & r_4u_4 & r_5u_5 & r_6u_6 & r_7u_7 & r_8u_8 & r_9u_9 \\ 0 & r_2u_1 & r_3u_2 & 0 & r_5u_4 & r_6u_5 & 0 & r_8u_7 & r_9u_8 \\ 0 & 0 & 0 & r_4u_2 & r_5u_3 & 0 & r_7u_5 & r_8u_6 & 0 \\ 0 & 0 & 0 & r_4u_1 & r_5u_2 & r_6u_3 & r_7u_4 & r_8u_5 & r_9u_6 \\ 0 & 0 & 0 & 0 & r_5u_1 & r_6u_2 & 0 & r_8u_4 & r_9u_5 \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{pmatrix}$$

Classification I



Classification II

In [0]:
```
#reshape data to fit model
X_train = X_train.reshape(60000,28,28,1)
X_test = X_test.reshape(10000,28,28,1)
```

In [7]:
```
#one-hot encode target column
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

y_train[0]
```

Out[7]:
```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)
```

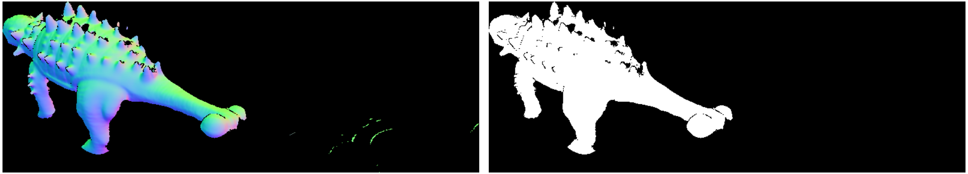
In [0]:
```
#create model
model = Sequential()

#add model layers
model.add(Conv2D(64, kernel_size=3, activation='relu', input_shape=(28,28,1)))
model.add(Conv2D(32, kernel_size=3, activation='relu'))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
```

In [0]:
```
#compile model using accuracy as a measure of model performance
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

In [10]:
```
#train model
model.fit(X_train, y_train,validation_data=(X_test, y_test), epochs=3)
```

Segmentation I

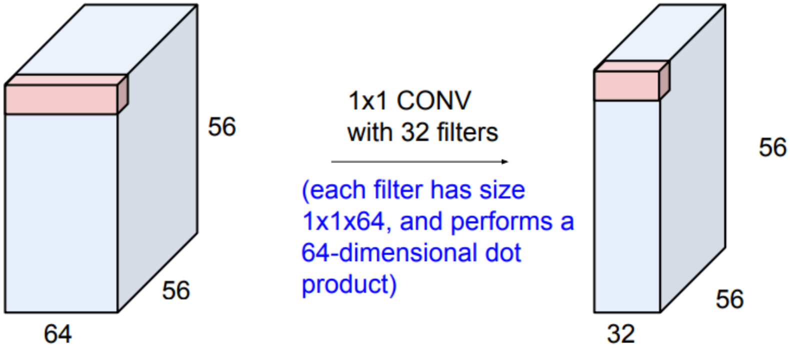
Segmentation II



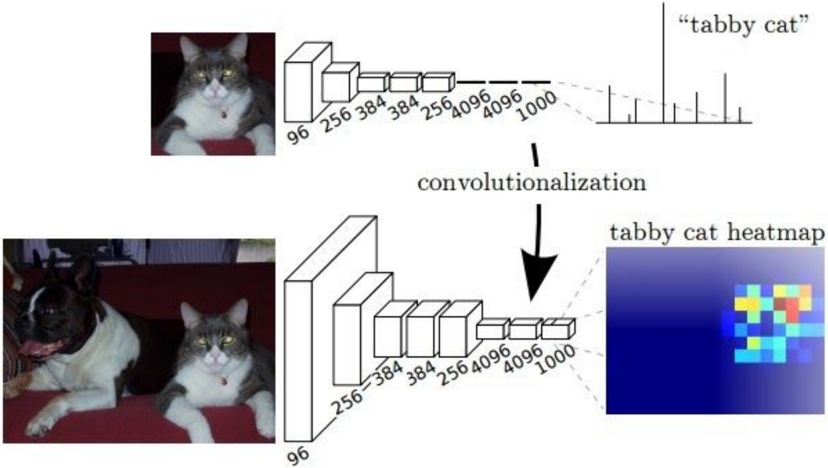
(a) input (b) output

FCN I

(btw, 1x1 convolution layers make perfect sense)



FCN II


"tabby cat"
96
256
384
384
256
4096
4096
1000
convolutionalization
tabby cat heatmap
96
256
384
384
256
4096
4096
1000

FCN III

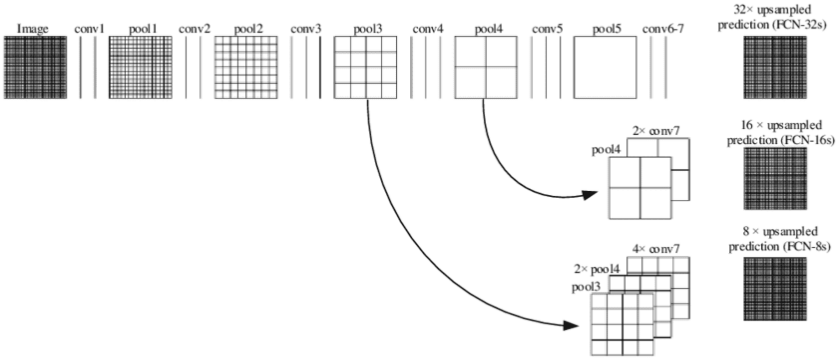
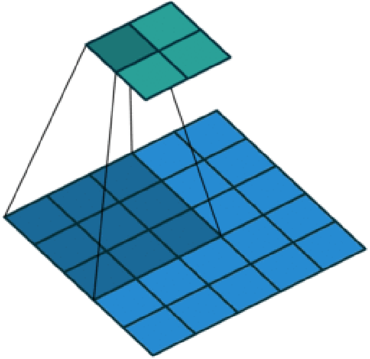
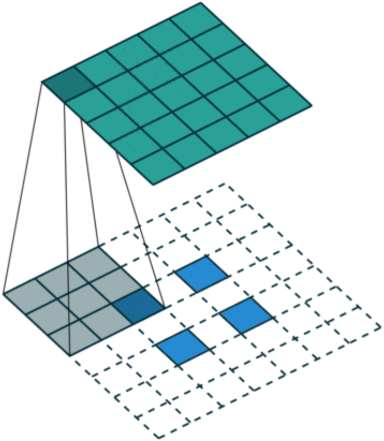


Image
conv1
pool1
conv2
pool2
conv3
pool3
conv4
pool4
conv5
pool5
conv6-7
32× upsampled prediction (FCN-32s)
2× conv7
pool4
16 × upsampled prediction (FCN-16s)
4× conv7
2× pool4
pool3
8 × upsampled prediction (FCN-8s)

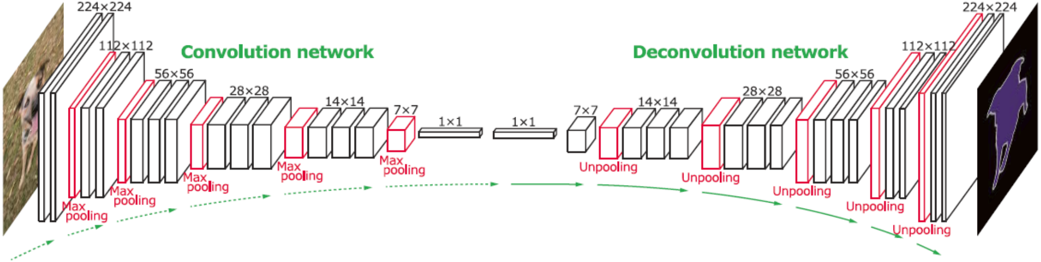
Deconvolution



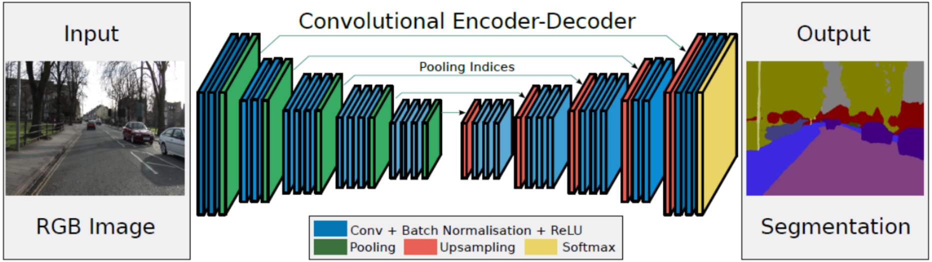
(a) convolution



(b) transposed

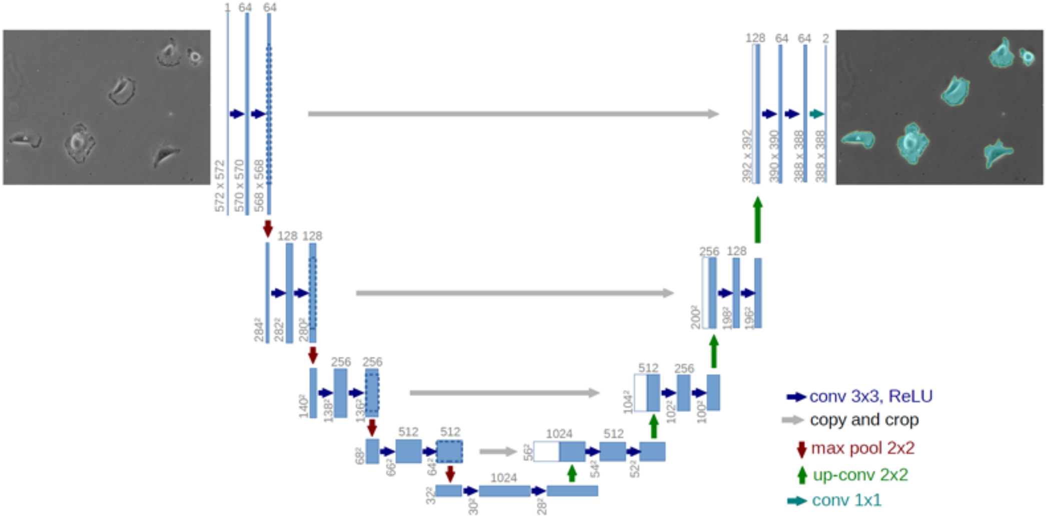

224×224
112×112
56×56
28×28
14×14
7×7
1×1
1×1
7×7
14×14
28×28
56×56
112×112
224×224
Convolution network
Deconvolution network
Max pooling
Max pooling
Max pooling
Max pooling
Max pooling
Unpooling
Unpooling
Unpooling
Unpooling
Unpooling

SegNet



SegNet: Encoder Decoder Architecture

U-Net I


1
64
64
128
128
256
256
512
512
1024
1024
512
256
128
64
2
572 x 572
570 x 570
568 x 568
392 x 392
390 x 390
388 x 388
388 x 388
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

U-Net II

```
conv2 = Conv2D(64, (3, 3), data_format=IMAGE_ORDERING,
               activation='relu', padding='same')(pool1)
conv2 = Dropout(0.2)(conv2)
conv2 = Conv2D(64, (3, 3), data_format=IMAGE_ORDERING,
               activation='relu', padding='same')(conv2)
pool2 = MaxPooling2D((2, 2), data_format=IMAGE_ORDERING)(conv2)

conv3 = Conv2D(128, (3, 3), data_format=IMAGE_ORDERING,
               activation='relu', padding='same')(pool2)
conv3 = Dropout(0.2)(conv3)
conv3 = Conv2D(128, (3, 3), data_format=IMAGE_ORDERING,
               activation='relu', padding='same')(conv3)

up1 = concatenate([UpSampling2D((2, 2), data_format=IMAGE_ORDERING)(
    conv3), conv2], axis=MERGE_AXIS)
conv4 = Conv2D(64, (3, 3), data_format=IMAGE_ORDERING,
               activation='relu', padding='same')(up1)
conv4 = Dropout(0.2)(conv4)
conv4 = Conv2D(64, (3, 3), data_format=IMAGE_ORDERING,
               activation='relu', padding='same')(conv4)
```

Comparison

Model	Year	Used Dataset	mAP as IoU
FCN-VGG16 [83]	2014	Pascal VOC 2012 [81]	62.2%
DeepLab[87]	2014	Pascal VOC 2012	71.6%
Deconvnet[90]	2015	Pascal VOC 2012	72.5%
U-Net[91]	2015	ISBI cell tracking challenge 2015	92% on PhC-U373 and 77.5% on DIC-HeLa dataset
DialatedNet [120]	2016	Pascal VOC 2012	73.9%
ParseNet [94]	2016	• ShiftFlow [76] • PASCAL- Context [128] • Pascal VOC 2012	40.4% 36.64% 69.8%
SegNet [93]	2016	• CamVid road scene segmentation [136] • SUN RGB-D indoor scene segmentation[137]	60.10% 31.84%
GCN[97]	2017	• PASCAL VOC 2012 •Cityscapes [138]	82.2% 76.9%
PSPNet [95]	2017	• PASCAL VOC 2012 • Cityscapes	85.4% 80.2%
FC-DenseNet103 [127]	2017	• CamVid road scene segmentation • Gatech[139]	66.9% 79.4%
EncNet [129]	2018	• Pascal VOC 2012 • Pascal Context	85.9% 51.7%

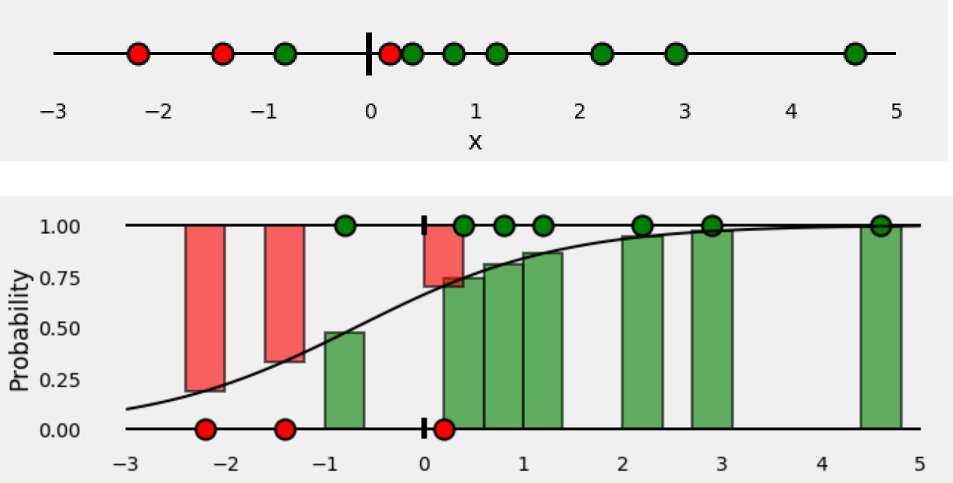
BCE I

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

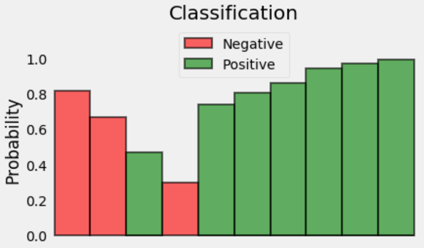
$$\text{(ground truth) } y_i = 1 \Leftrightarrow \text{true geometry}, y_i = 0 \Leftrightarrow \text{artefact}$$

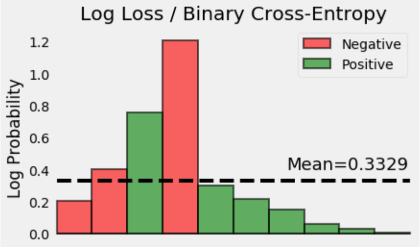
$$\text{(prediction) } p(y_i) \in \langle 0, 1 \rangle \Leftrightarrow \text{returned by ML system}$$

BCE II



BCE III


Classification
Negative
Positive
Probability
1.0
0.8
0.6
0.4
0.2
0.0


Log Loss / Binary Cross-Entropy
Negative
Positive
Log Probability
Mean=0.3329
1.2
1.0
0.8
0.6
0.4
0.2
0.0

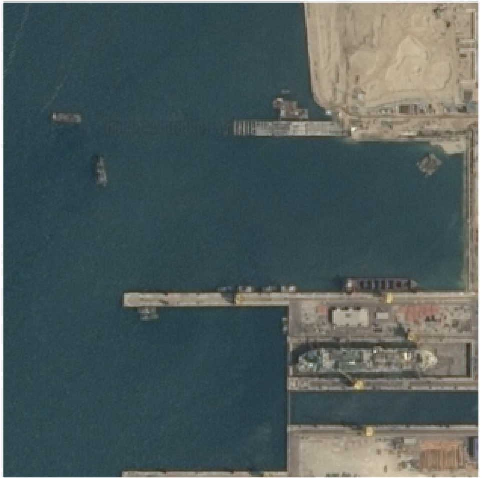
BCE Math

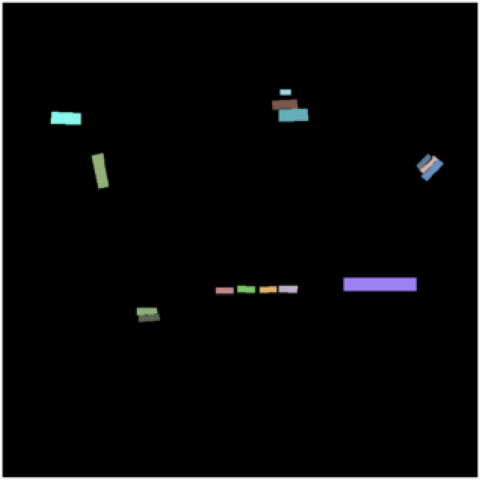
$$ENTROPY = H(q) = -\sum_{c=1}^{C} q(c) \cdot log(q(c))$$

$$BCE = H_p(q) = -\sum_{c=1}^{C} q(c) \cdot log(p(c))$$

$$H_p(q) - H(q) \geq 0$$

Class Imbalane





Confusion Matrix

		Prediction	
		True	Artefact
$CM =$ Actual	True	18508 (*TP*)	53 (*FN*)
	Artefact	112 (*FP*)	166 (*TN*)

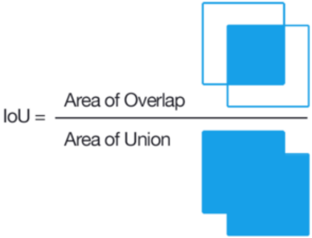
$$Accuracy(CM) = \frac{TP + TN}{TP + TN + FP + FN} \sim \mathbf{0.991}$$

$$TPR(CM)\ (Recall) = \frac{TP}{TP + FN} \sim 0.997 \mid PPV(CM)\ (Precision) = \frac{TP}{TP + FP} \sim 0.994$$

$$Balanced\ Accuracy(CM) = \frac{TPR + TNR}{2} \sim \mathbf{0.80}$$

$$F_1 = 2 \cdot \frac{PPV \cdot TPR}{(PPV + TPR)} \sim 0.995 \mid FOR(CM) = \frac{FN}{FN + TN} \sim 0.24$$

IoU



$$IOU = J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

$$IOU = \left(\frac{TP}{TP + FP + FN} + \frac{TN}{TN + FP + FN} \right) /2 =$$

$$\left(\frac{18508}{18508 + 112 + 53} + \frac{166}{166 + 112 + 53} \right) /2 \sim (\mathbf{0.991} + \mathbf{0.502})/\mathbf{2} \sim \mathbf{0.746}$$

Article Sources

- LONG, J., SHELHAMER, E., AND DARRELL, T.
  Fully convolutional networks for semantic segmentation, 2014.
- NOH, H., HONG, S., AND HAN, B.
  Learning deconvolution network for semantic segmentation, 2015.
- RONNEBERGER, O., FISCHER, P., AND BROX, T.
  U-Net: convolutional networks for biomedical image segmentation.
  vol. 9351, pp. 234–241.
- SULTANA, F., SUFIAN, A., AND DUTTA, P.
  Evolution of image segmentation using deep convolutional neural network: A survey.
  Knowledge-Based Systems 201-202 (Aug 2020), 106062.

Web Sources

- http://cs231n.stanford.edu/
- https://nanonets.com/blog/semantic-image-segmentation-2020/
- https://divamgupta.com/image-segmentation/2019/06/06/deep-learning-semantic-segmentation-keras.html
- https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d
- https://towardsdatascience.com/dropout-on-convolutional-layers-is-weird-5c6ab14f19b2
- https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6
- https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2
- https://calebrob.com/ml/2018/09/11/understanding-iou.html
- https://github.com/ColinShaw/python-keras-encoder-decoder-unet
- https://github.com/gajdosech2/pc-filtering