Object Localization

- For each photo input, the algorithm will output 5 class labels denoted

$$c_i, i = 1, 2, ...5$$

in decreasing order of confidence.

- The algorithm will also output 5 bounding boxes denoted

$$b_i, i = 1, 2, ..., 5$$

for each class label.

- The ground truth labels are for the image

$$C_k, k = 1, 2, ..., n$$

class labels.

Object Localization

- For each ground truth class label C_k, the ground truth bounding boxes are

$$B_{km}, m = 1, 2, ..., M_k$$

  where M_k is the number of instances of the k^{th} object in the current image.
- Let $d(c_i, C_k) = 0$ if $c_i = C_k$ and 1 otherwise.

Object Localization

- The function that we are trying to minimize is

$$e = \frac{1}{n}\sum(min_i min_m max d(c_i, C_k), f(b_i, B_{km}))$$

where d is 0 if your algorithm predicts the same class as the bounded class label and $f(b_i, Bkm) = 0$ if your algorithm's bounding box overlaps the ground truth bounding box by more than 50%.

- For each input image, your algorithm must produce a set of annotations

$$(c_i, s_i, b_i)$$

where c_i are the class labels, s_i are the confidence scores, and b_i are the bounding boxes.

Bounding Box Example

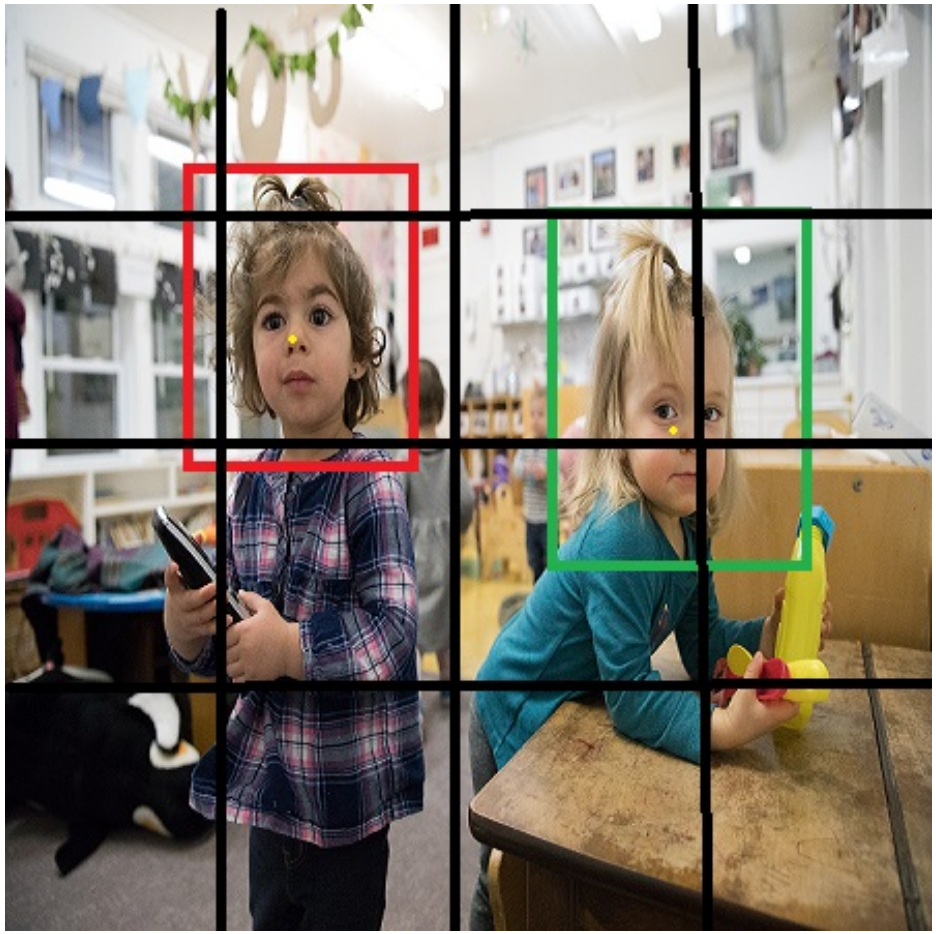


Figure: For each grid cell, we will attempt to predict a class object. If a class object is found, identify the midpoint of that class object and then draw the bounding box.

Inception V3

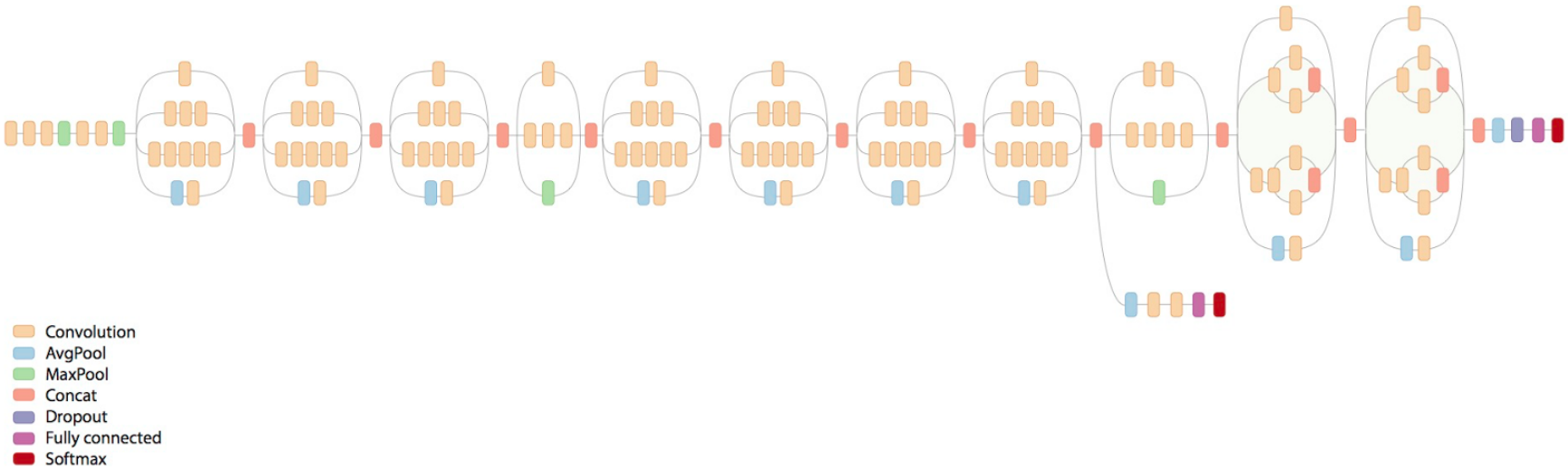


Figure: Inception's structure has 22 parameterized layers, 5 pooling layers, and its inception modules that bring the total layer count to 100.

Inception Modules

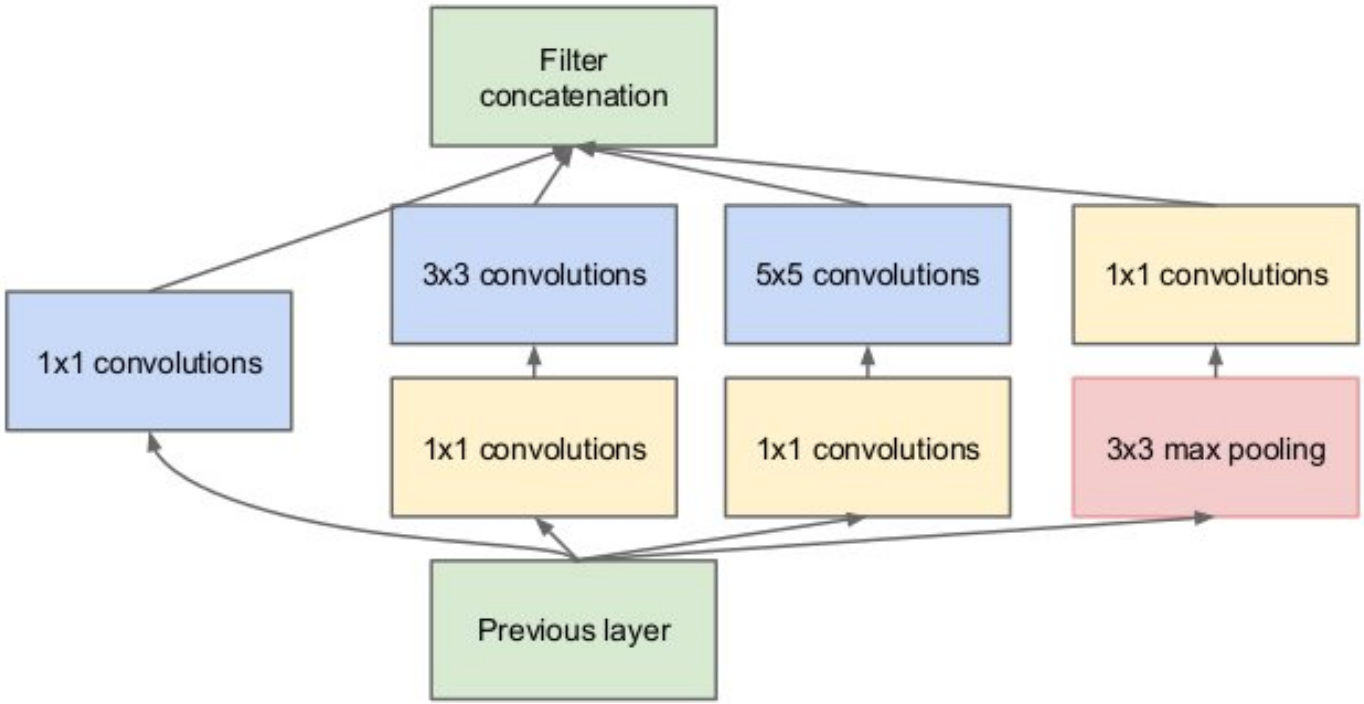


Figure: The inception module allows for multiple convolutional filters to retain both local and high abstraction features while reducing the dimensionality of the network.

Inception Table

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture

Figure:

Calculating the Operations

- In order to demonstrate how to calculate the total number of parameters and the dimensionality reduction employed by the inception module, we can employ the following formula:

$$Number \quad of \quad Operations = F \cdot D \cdot I \cdot M$$

where F is the convolutional filter size, D is the matrix dimension of the input, I is the the number of feature map inputs and M is the number of feature map outputs.

Operations Example

- For a 5x5 convolutional filter, with $D = 28$ x 28, $I = 192$, $M = 32$, the total number of operations would be

$$(5 \cdot 5)(28 \cdot 28)(192)(32) = 120,422,000 \quad \textit{operations}$$

- For a 1x1 convolutional filter, with $D = 28$ x 28, $I = 192$ and $M = 16$

$$(1 \cdot 1)(28 \cdot 28)(192)(16) = 2,408,448 \quad \textit{operations}$$

Operations Example

- If we were to place the 1x1 convolutional filter in front of the 5x5 convolutional filter, and calculate the same size feature map as the aforementioned 5x5 filter, we would reduce the number of feature map inputs to 16 giving the following calculation:

$$(1 \cdot 1)(28 \cdot 28)(192)(16) + ((5 \cdot 5)(28 \cdot 28)(16)(32)$$

$$2,408,448 + 10,035,200 = 12,443,648 \quad \textit{operations}$$

- By placing the 1x1 convolutional filter in front of the 5x5 convolutional filter, the we maintain the same 28x28x32 output volume but only using $\frac{12,443,648}{120,422,000} = 10.333\%$ of the original 5x5 filter operations.

Batch Normalization Algorithm

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Figure: Inception V3 utilizes the Batch Normalization algorithm from the paper *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift* by Ioffe and Szegedy, 2015.

ADAM Optimization Algorithm

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Figure: Inception V3 uses the ADAM algorithm from the *ADAM: A Method For Stochastic Optimization* by Kingma and Ba, 2015.

Minimum Error Algorithm

```
min_error_list = []
for prediction_box in [p_0, p_1, p_2]:
    max_error_list = []
    for ground_truth_box in [g_0, g_1]:
        if label(prediction_box) == label(ground_truth_box):
            d = 0
        else:
            d = 1
        if overlap(prediction_box, ground_truth_box) > 0.5:
            f = 0
        else:
            f = 1
        max_error_list.append(max(d,f))   # the first max
    min_error_list.append(min(max_list))  # the first min

return min(min_error_list)  # the second min
```

ImageNet Data Set

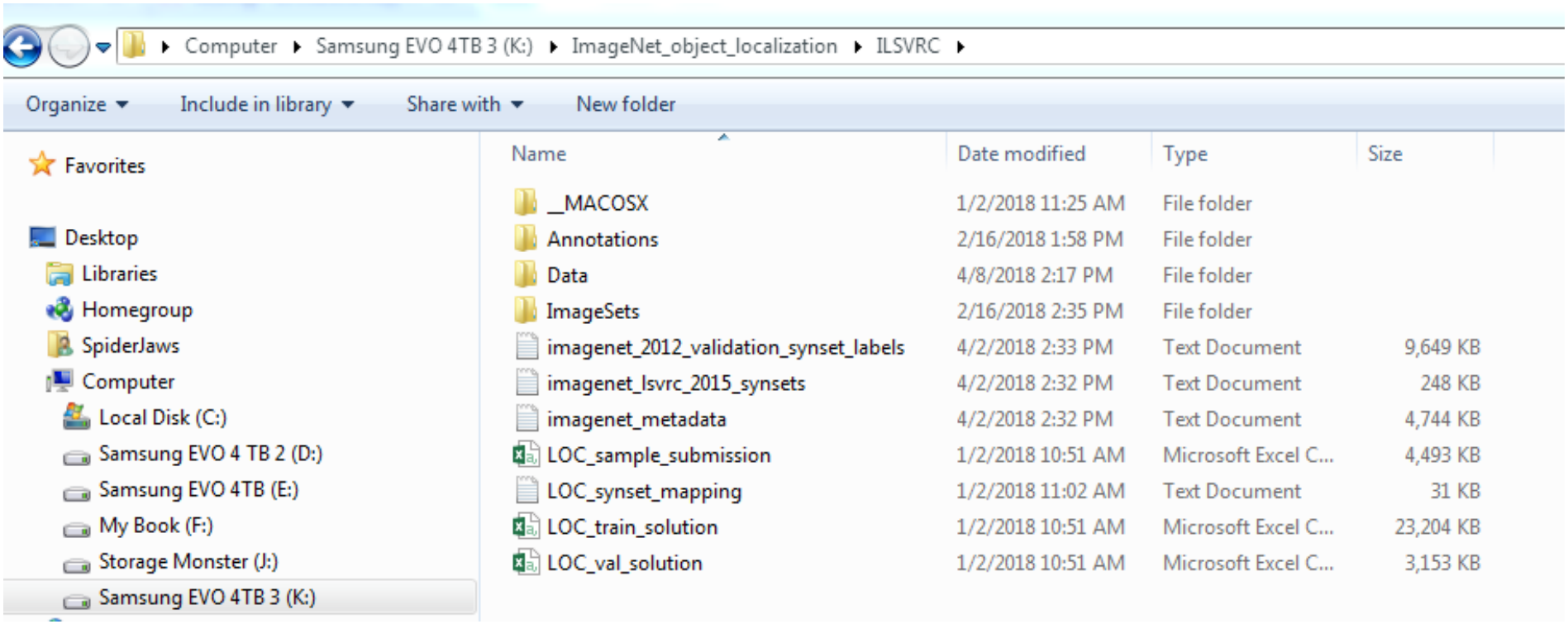


Figure: The ImageNet Data set consists of 150,000 training images, 50,000 validation images, and 40,000 test images. Each image is hand-annotated with class label from the 1000 synsets, and a ground truth bounding box.

TFRecords Conversion

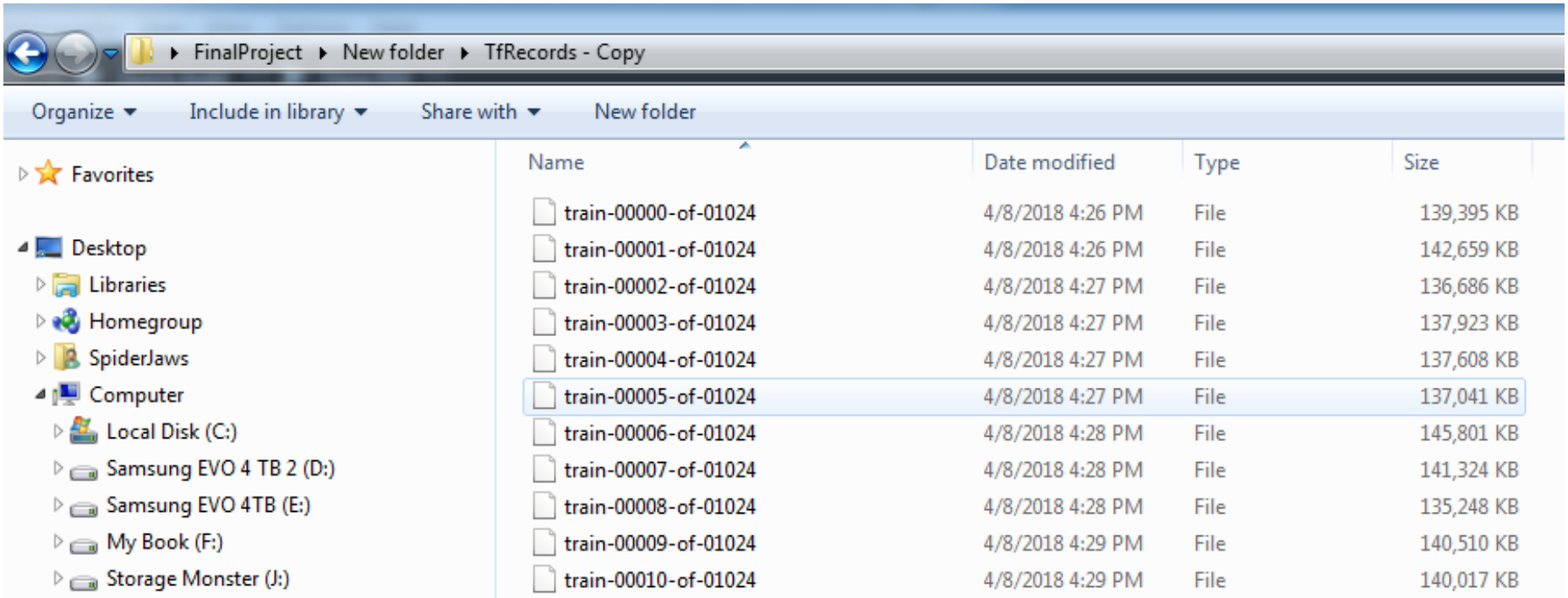


Figure: The TFRecords are created by using Protocol buffers that attach the class label and ground truth bounding box that reside in a separate XML. The TFRecords also have human readable annotations for class labels rather than the ImageNet's proprietary class coding.

Image Preprocessing

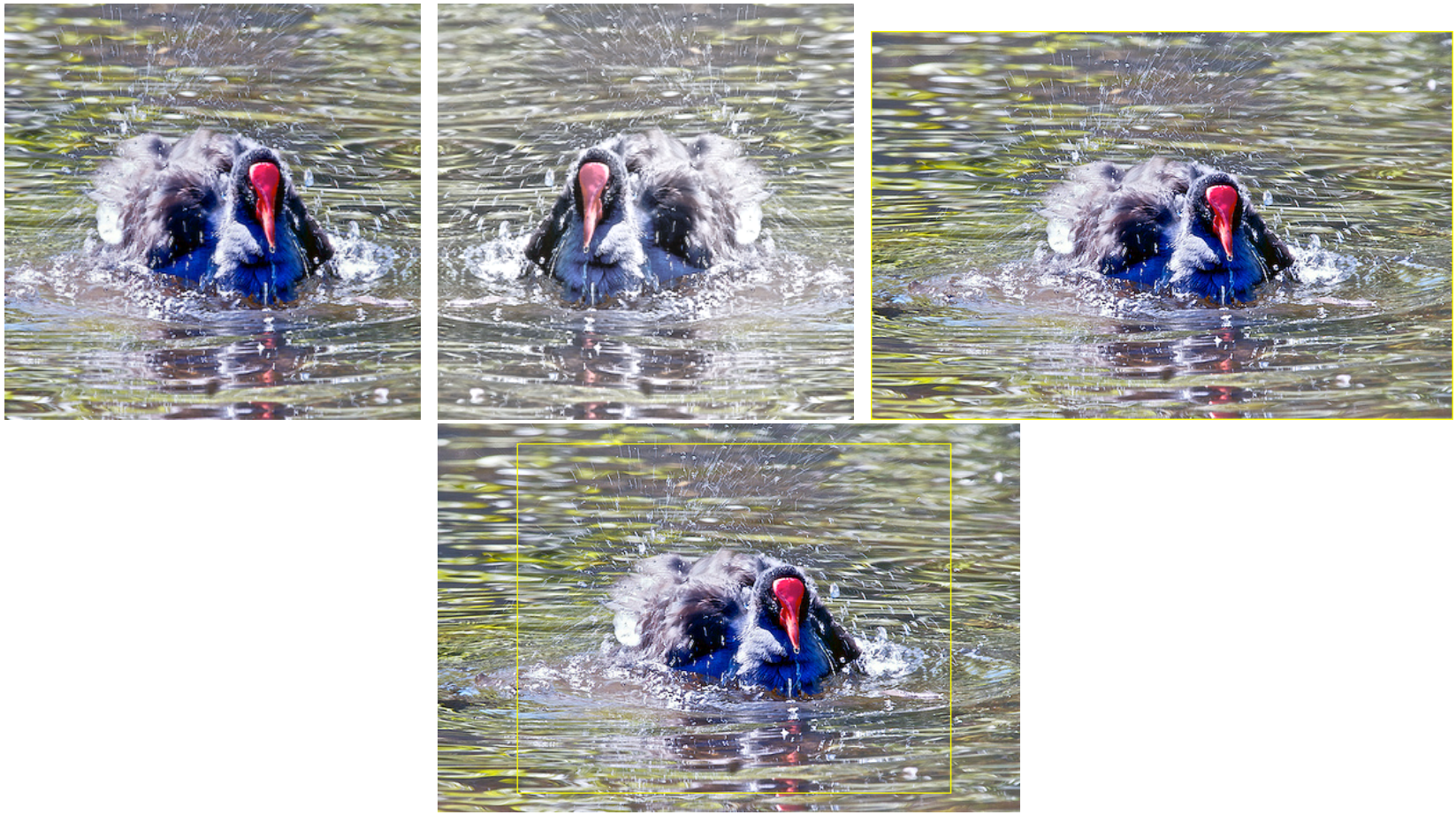


Figure: **Top Left:** Cropped and Resized **Top Right:** Flipped with random adjustments to brightness and hue **Bottom Left:** Stretched image with redrawn bounding box around edges **Bottom Right:** Resized image with resized bounding box

Model Training and Evaluation

Figure: After 632 global steps, the classification loss as measured by cross-entropy was 4.5 while the bounding box loss was 2.36 for a total loss of 6.86.

Model Training and Evaluation

```
INFO:tensorflow:Starting evaluation at 2018-04-
INFO:tensorflow:Restoring parameters from C:\Us
ct\IRAIN2\model.ckpt-626
INFO:tensorflow:Evaluation [6/60]
INFO:tensorflow:Evaluation [12/60]
INFO:tensorflow:Evaluation [18/60]
INFO:tensorflow:Evaluation [24/60]
INFO:tensorflow:Evaluation [30/60]
INFO:tensorflow:Evaluation [36/60]
INFO:tensorflow:Evaluation [42/60]
INFO:tensorflow:Evaluation [48/60]
INFO:tensorflow:Evaluation [54/60]
INFO:tensorflow:Evaluation [60/60]
eval/Accuracy[0.544]eval/Recall_5[0.75516665]
```

Figure: After evaluating 60 evaluation sets, the top-1 accuracy was 54.4% while the Top-5 accuracy was 75.5%

How does that compare to other models?

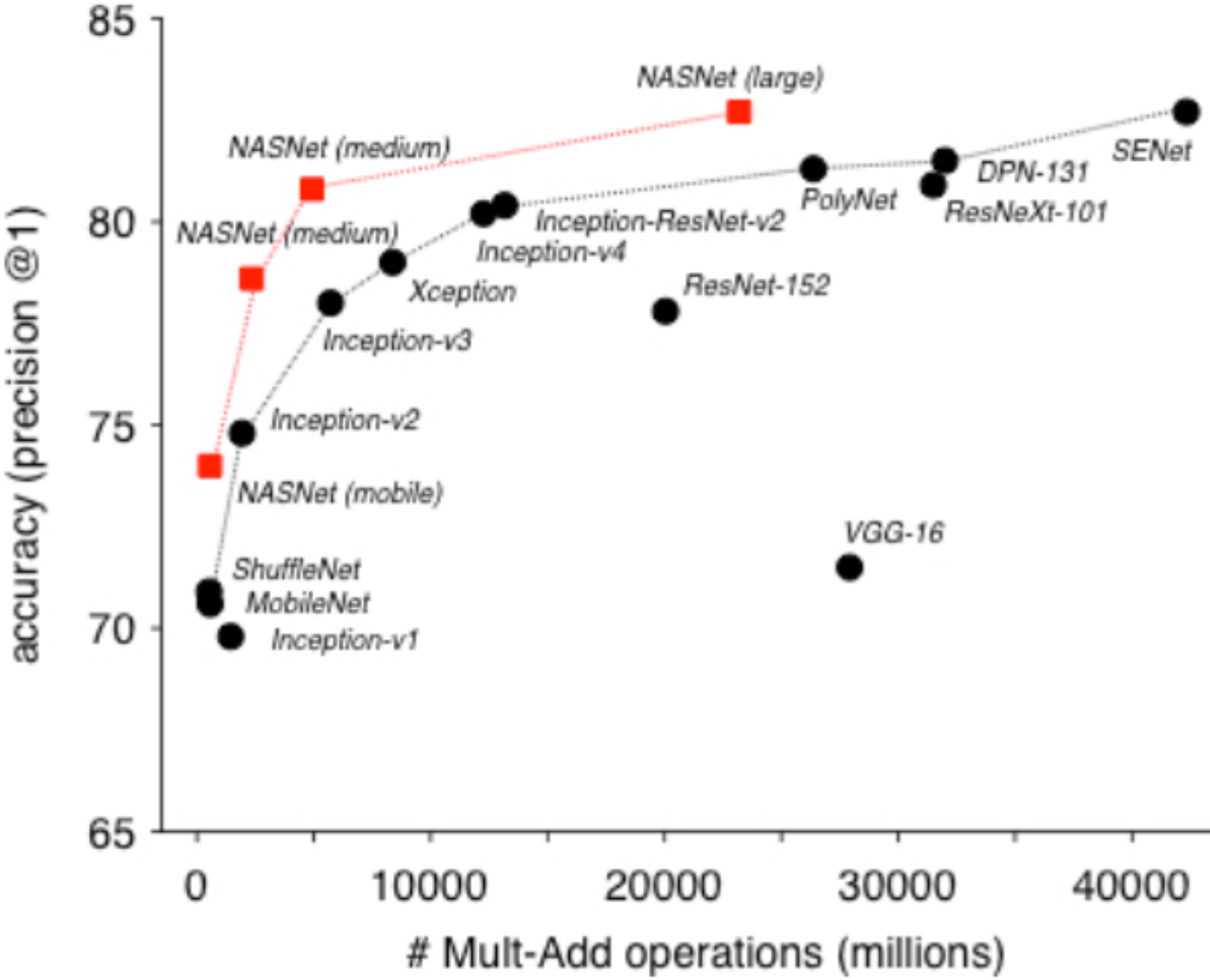


Figure: When Google trained and evaluated Inception V3, they acheived a top1 accuracy of 78% and a top-5 accuracy of 93.9%

Image Classifier Output

```
swing (score = 0.19141)
shovel (score = 0.15934)
parallel bars, bars (score = 0.15499)
horizontal bar, high bar (score = 0.03545)
chime, bell, gong (score = 0.03447)
```

Image Classifier Output

```
apron (score = 0.92674)
paintbrush (score = 0.02684)
pajama, pyjama, pj's, jammies (score = 0.01278)
spatula (score = 0.00921)
handkerchief, hankie, hanky, hankey (score = 0.00231)
```

Image Classifier Output

```
home theater, home theatre (score = 0.64070)
entertainment center (score = 0.05868)
television, television system (score = 0.05760)
tricycle, trike, velocipede (score = 0.01467)
vacuum, vacuum cleaner (score = 0.01444)
```

Image Classifier Output

```
seashore, coast, seacoast, sea-coast (score = 0.41154)
umbrella (score = 0.05807)
sarong (score = 0.04080)
maillot, tank suit (score = 0.03249)
maillot (score = 0.02926)
```

Image Classifier Output

```
stage (score = 0.25885)
loudspeaker, speaker, speaker unit, loudspeaker system, speake
laptop, laptop computer (score = 0.05895)
computer keyboard, keypad (score = 0.04785)
mouse, computer mouse (score = 0.04469)
```

Image Classifier Output

```
horse cart, horse-cart (score = 0.93709)
oxcart (score = 0.00455)
barn (score = 0.00231)
thresher, thrasher, threshing machine (score = 0.00176)
ox (score = 0.00145)
```

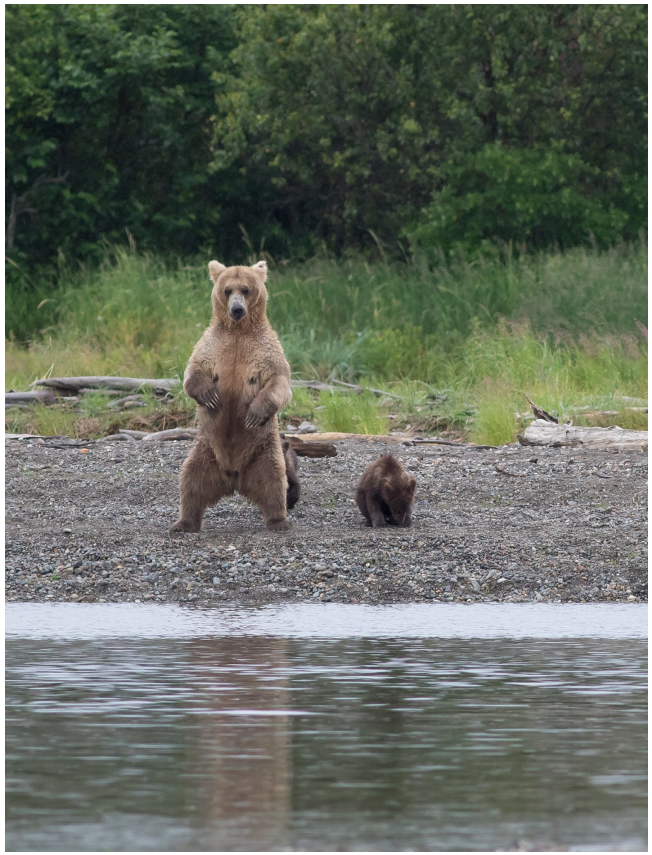
Image Classifier Output

```
vulture (score = 0.65422)
little blue heron, Egretta caerulea (score = 0.05282)
European gallinule, Porphyrio porphyrio (score = 0.04539)
black grouse (score = 0.02519)
bald eagle, American eagle, Haliaeetus leucocephalus (score =
```

Image Classifier Output

```
bald eagle, American eagle, Haliaeetus leucocephalus (score =
kite (score = 0.01935)
vulture (score = 0.00074)
velvet (score = 0.00050)
black grouse (score = 0.00048)
```

Image Classifier Output



```
brown bear, bruin, Ursus arctos (score = 0.95635)
Airedale, Airedale terrier (score = 0.00609)
Irish water spaniel (score = 0.00138)
teddy, teddy bear (score = 0.00125)
Irish terrier (score = 0.00077)
```

Image Classifier Output

```
brown bear, bruin, Ursus arctos (score = 0.95426)
bison (score = 0.00335)
bottlecap (score = 0.00117)
lesser panda, red panda, panda, bear cat, cat bear, Ailurus fu
hyena, hyaena (score = 0.00043)
```

Image Classifier Output

```
brown bear, bruin, Ursus arctos (score = 0.98223)
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus (s
Irish water spaniel (score = 0.00032)
bottlecap (score = 0.00025)
American black bear, black bear, Ursus americanus, Euarctos am
```

Image Classifier Output

```
brown bear, bruin, Ursus arctos (score = 0.29495)
lesser panda, red panda, panda, bear cat, cat bear, Ailurus fu
Norwich terrier (score = 0.02791)
Australian terrier (score = 0.01975)
chow, chow chow (score = 0.01757)
```

Image Classifier Output

```
Lunatic (score = 1)
Windsor tie (score = 0)
academic gown, academic robe, judge's robe (score = 0)
suit, suit of clothes (score = 0)
mortarboard (score = 0)
```