Iterative, Multiplayer Research
Accounts Needed for Pt. 2, Competition

kaggle.com

wandb.ai

colab.research.google.com
Weights & Biases

The W&B Course

www.wandb.courses
The Goal

Table 3: Detection results on PASCAL VOC 2007 test set. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07+12": union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. This number was reported in [2]; using the repository provided by this paper, this result is higher (68.1).

<table>
<thead>
<tr>
<th>method</th>
<th># proposals</th>
<th>data</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>2000</td>
<td>07</td>
<td>66.9</td>
</tr>
<tr>
<td>SS</td>
<td>2000</td>
<td>07+12</td>
<td>70.0</td>
</tr>
<tr>
<td>RPN+VGG, unshared</td>
<td>300</td>
<td>07</td>
<td>66.5</td>
</tr>
<tr>
<td>RPN+VGG, shared</td>
<td>300</td>
<td>07+12</td>
<td>69.9</td>
</tr>
<tr>
<td>RPN+VGG, shared</td>
<td>300</td>
<td>07+12</td>
<td>73.2</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>method</th>
<th># proposals</th>
<th>data</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>2000</td>
<td>12</td>
<td>65.7</td>
</tr>
<tr>
<td>SS</td>
<td>2000</td>
<td>07+12</td>
<td>68.4</td>
</tr>
<tr>
<td>RPN+VGG, shared</td>
<td>300</td>
<td>12</td>
<td>67.0</td>
</tr>
<tr>
<td>RPN+VGG, shared</td>
<td>300</td>
<td>07+12</td>
<td>70.4</td>
</tr>
<tr>
<td>RPN+VGG, shared</td>
<td>300</td>
<td>COCO+07+12</td>
<td>78.9</td>
</tr>
</tbody>
</table>

Table 5: Timing (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. "Region-wise" includes NMS, pooling, fully-connected, and softmax layers. See our released code for the profiling of running time.

<table>
<thead>
<tr>
<th>model</th>
<th>system</th>
<th>conv</th>
<th>proposal</th>
<th>region-wise</th>
<th>total</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG</td>
<td>SS + Fast R-CNN</td>
<td>146</td>
<td>1510</td>
<td>174</td>
<td>1830</td>
<td>0.5 fps</td>
</tr>
<tr>
<td></td>
<td>RPN + Fast R-CNN</td>
<td>141</td>
<td>10</td>
<td>47</td>
<td>198</td>
<td>5 fps</td>
</tr>
<tr>
<td>ZF</td>
<td>RPN + Fast R-CNN</td>
<td>31</td>
<td>3</td>
<td>25</td>
<td>59</td>
<td>17 fps</td>
</tr>
</tbody>
</table>
0000 p1 DRE 3.11 - not super eval
meg threshold 0.21
2L 300N BLSTM (BasicLSTM)
LOD
sigmoid
AdamOptimizer
100 frames
dropout 1.0
zero input and label
Log(1+1.0)
100330 training, 2000 CV
model:
weights20170224-005946_v10.1419
ps, loss .1419, epoch 40 [task0]
MEAN IBM SDR GAIN: 2.324 -
with 0.15 threshold during cluster
STD IBM SDR GAIN: 2.276
MEAN IBM SDR GAIN: 2.110 -
with 0.32 threshold during cluster
STD IBM SDR GAIN: 2.254

0001 p2 DRE 3.23
meg threshold 0.12
2L 300N BLSTM_clean (LSTM & many
rewards) - note, this was the
massive model rewrite
200
sigmoid
AdamOptimizer
150 frames
dropout 1.0
zero input and label
log(1+1.0)
10330 training, 2000 CV
model:
weights20170224-032054_v10.1418
ps, loss .1418, epoch 40 [task0]
MEAN IBM SDR GAIN: 2.056 -
with 0.15 threshold during cluster
STD IBM SDR GAIN: 2.214
MEAN IBM SDR GAIN: 2.068 -
with 0.32 threshold during cluster
STD IBM SDR GAIN: 2.205
MEAN IBM SDR GAIN: 2.108 - with
fancy best SDR of the two system
(eval_stdrl.py)
STD IBM SDR GAIN: 4.114
MEAN IBM SDR GAIN: 5.915 -
using EXACT script measuring both
voices gain (SUPER_EVAL)
STD IBM SDR GAIN: 4.349
<table>
<thead>
<tr>
<th>Experiment Name</th>
<th>Created</th>
<th>train_loss</th>
<th>valid_loss</th>
<th>acc</th>
<th>traffic_acc</th>
<th>road_acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>best car acc (50% data)</td>
<td>2021-04-14</td>
<td>0.5375041962</td>
<td>0.442730248</td>
<td>0.8823291659</td>
<td>0.8663836718</td>
<td>0.9399003386</td>
</tr>
<tr>
<td>best traffic acc (50% data)</td>
<td>2021-04-14</td>
<td>0.4919361174</td>
<td>0.4202951491</td>
<td>0.8879730701</td>
<td>0.8718349934</td>
<td>0.9439761043</td>
</tr>
<tr>
<td>best overall IOU (20% data)</td>
<td>2021-04-14</td>
<td>0.5095784068</td>
<td>0.465856516</td>
<td>0.8725891709</td>
<td>0.8592621684</td>
<td>0.9359762073</td>
</tr>
<tr>
<td>major-sweep-196</td>
<td>2021-01-31</td>
<td>0.5705417991</td>
<td>0.4875227213</td>
<td>0.8698127866</td>
<td>0.8570486426</td>
<td>0.9454026222</td>
</tr>
<tr>
<td>swept-sweep-164</td>
<td>2021-01-31</td>
<td>0.5535062551</td>
<td>0.4849829972</td>
<td>0.8701210618</td>
<td>0.8567070365</td>
<td>0.9204238057</td>
</tr>
<tr>
<td>silver-sweep-139</td>
<td>2021-01-31</td>
<td>0.563354373</td>
<td>0.5251665628</td>
<td>0.871628046</td>
<td>0.846842885</td>
<td>0.9262287617</td>
</tr>
<tr>
<td>laced-sweep-115</td>
<td>2021-01-31</td>
<td>0.5277443528</td>
<td>0.5124291778</td>
<td>0.8705932498</td>
<td>0.8521651027</td>
<td>0.9389513731</td>
</tr>
<tr>
<td>eager-sweep-97</td>
<td>2021-01-31</td>
<td>0.5488699675</td>
<td>0.5005864501</td>
<td>0.8738754392</td>
<td>0.8612990975</td>
<td>0.913561523</td>
</tr>
<tr>
<td>rich-sweep-88</td>
<td>2021-01-31</td>
<td>0.5587444901</td>
<td>0.5211353302</td>
<td>0.8785927892</td>
<td>0.8512274623</td>
<td>0.9295567274</td>
</tr>
<tr>
<td>hopeful-sweep-33</td>
<td>2021-01-31</td>
<td>0.503461957</td>
<td>0.4650281966</td>
<td>0.8706912994</td>
<td>0.8560319543</td>
<td>0.9387732744</td>
</tr>
<tr>
<td>autumn-sweep-24</td>
<td>2021-01-31</td>
<td>0.5777919888</td>
<td>0.500880897</td>
<td>0.8755427003</td>
<td>0.8561192751</td>
<td>0.9181208611</td>
</tr>
<tr>
<td>decent-sweep-21</td>
<td>2021-01-31</td>
<td>0.5714729428</td>
<td>0.4979581237</td>
<td>0.8745227456</td>
<td>0.8490597606</td>
<td>0.9463140965</td>
</tr>
<tr>
<td>vague-sweep-5</td>
<td>2021-01-31</td>
<td>0.6230331063</td>
<td>0.473508656</td>
<td>0.8732874393</td>
<td>0.8601382971</td>
<td>0.9297611117</td>
</tr>
<tr>
<td>Second best acc</td>
<td>2021-01-31</td>
<td>0.4194990396</td>
<td>0.4509623024</td>
<td>0.8873019218</td>
<td>0.8705806732</td>
<td>0.9445936084</td>
</tr>
<tr>
<td>Experiment Name</td>
<td>Created</td>
<td>train_loss</td>
<td>valid_loss</td>
<td>acc</td>
<td>traffic_acc</td>
<td>modacc</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------</td>
<td>------------</td>
<td>------------</td>
<td>-----</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td>best car acc (95% data)</td>
<td>2021-04-14</td>
<td>0.0370401962</td>
<td>0.0427373028</td>
<td>0.8653391656</td>
<td>0.6663951718</td>
<td>0.8989930306</td>
</tr>
<tr>
<td>best traffic acc (95% data)</td>
<td>2021-04-14</td>
<td>0.4919081174</td>
<td>0.4202001340</td>
<td>0.8028730701</td>
<td>0.8162439004</td>
<td>0.8469810104</td>
</tr>
<tr>
<td>best overall (95% data)</td>
<td>2021-04-14</td>
<td>0.0286784968</td>
<td>0.4188190017</td>
<td>0.8688322547</td>
<td>0.4832620029</td>
<td>0.6886630697</td>
</tr>
<tr>
<td>(num sweeps=10)</td>
<td>2021-01-31</td>
<td>0.5278471701</td>
<td>0.4978227230</td>
<td>0.6232922567</td>
<td>0.4568823079</td>
<td>0.5211659268</td>
</tr>
<tr>
<td>(num sweeps=5)</td>
<td>2021-01-31</td>
<td>0.5633535273</td>
<td>0.5212502678</td>
<td>0.8573465428</td>
<td>0.5142917219</td>
<td>0.5215165426</td>
</tr>
<tr>
<td>(num sweeps=2)</td>
<td>2021-01-31</td>
<td>0.6297443026</td>
<td>0.5219333002</td>
<td>0.6987440021</td>
<td>0.6213393002</td>
<td>0.6987440021</td>
</tr>
<tr>
<td>(num sweeps=2)</td>
<td>2021-01-31</td>
<td>0.5254151867</td>
<td>0.4985281164</td>
<td>0.5777918668</td>
<td>0.5668808007</td>
<td>0.5777918668</td>
</tr>
<tr>
<td>(num sweeps=2)</td>
<td>2021-01-31</td>
<td>0.5374739228</td>
<td>0.4978551237</td>
<td>0.6152310363</td>
<td>0.4732082004</td>
<td>0.6152310363</td>
</tr>
<tr>
<td>(num sweeps=2)</td>
<td>2021-01-31</td>
<td>0.4594933006</td>
<td>0.4569832502</td>
<td>0.6359493306</td>
<td>0.4569832502</td>
<td>0.6359493306</td>
</tr>
</tbody>
</table>
Ben Sherman (wandb.com)

See strange results in cell 28?

Not sure what's going on here. Can you take a look?

pedestrian-detect-ptrl-2022-03-066-be420fba-5... (146K)
Ben Sherman (wandb.com)

See strange results in cell 28?

Not sure what's going on here. Can you take a look?

pedestrian-detect-ptl-2022-03-066-be420fba-5... (146K)
THIS IS FINE.
Just Stop
Three principles of an ideal ML workflow

Rapidly iterate
- to continuously refine and optimize models

Reproduce
- to reduce key-person dependencies

Collaborate
- to ensure knowledge transfer across the organization
Three principles of an ideal ML workflow

- **Rapidly iterate**
  to continuously refine and optimize models

- **Reproduce**
  to reduce key-person dependencies

- **Collaborate**
  to ensure knowledge transfer across the organization
A system of record for all ML workflows
A system of record for all ML workflows
A system of record for all ML workflows

Get started in 60 seconds

```python
!pip install wandb  # Install W&B
wandb.init()       # Start experiment
wandb.log(metrics) # Log metrics + more!
```
Yes, you really can get started in 60 seconds.
A system of record for wandb.Image
A system of record for `wandb.Object3D`
### A system of record for \texttt{wandb.Audio}

<table>
<thead>
<tr>
<th>Whale song</th>
<th>Audio features</th>
<th>Spectrogram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Whale song" /></td>
<td><img src="image2" alt="Audio features" /></td>
<td><img src="image3" alt="Spectrogram" /></td>
</tr>
<tr>
<td><img src="image4" alt="Whale song" /></td>
<td><img src="image5" alt="Audio features" /></td>
<td><img src="image6" alt="Spectrogram" /></td>
</tr>
</tbody>
</table>

00:00/00:46

00:00/00:07
A system of record for wandb.Video
A system of record for wandb.plots.POS
A system of record for all types of data!

wandb.Image
wandb.Object3D
wandb.log(wandb.Molecule)
wandb.Video
wandb.Html

For a complete set of types, visit docs.wandb.ai
Ask questions about your data and models
Tables – DataFrames with rich media support

Ask questions about your data (and models)

Filter
df.loc[...]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Industry</th>
<th>Shares</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSFT</td>
<td>Tech</td>
<td>100</td>
</tr>
<tr>
<td>GOOG</td>
<td>Tech</td>
<td>50</td>
</tr>
<tr>
<td>TSLA</td>
<td>Automotive</td>
<td>150</td>
</tr>
</tbody>
</table>

Industry = "Tech"
Shares < 100

GOOG Tech 50
Ask questions about your data (and models)

**Tables - DataFrames with rich media support**

Filter

```
In [1]: df.loc[...]  
```

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Industry</th>
<th>Shares</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSFT</td>
<td>Tech</td>
<td>100</td>
</tr>
<tr>
<td>GOOG</td>
<td>Tech</td>
<td>50</td>
</tr>
<tr>
<td>TSLA</td>
<td>Automotive</td>
<td>150</td>
</tr>
</tbody>
</table>

```
Industry = "Tech"  
Shares < 100       
```

```
In [2]: df.groupby(...).agg(...)  
```

**Split-Apply-Combine**

```
df.groupby(...).agg(...)  
```

Diagram showing the process of splitting, applying, and combining data.
Tables – DataFrames with rich media support

Ask questions about your data (and models)

Filter
df.loc[...]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Industry</th>
<th>Shares</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSFT</td>
<td>Tech</td>
<td>100</td>
</tr>
<tr>
<td>GOOG</td>
<td>Tech</td>
<td>50</td>
</tr>
<tr>
<td>TSLA</td>
<td>Automotive</td>
<td>150</td>
</tr>
</tbody>
</table>

Industry = "Tech"
Shares < 100

GOOG  Tech  50

Split-Apply-Combine
df.groupby(...).agg(...)

Facet / Plot
df.plot(...)
How do I ask questions about non-tabular data?
## Tables – DataFrames with rich media support

Ask questions about your data (and models)

### Classic DataFrames

<table>
<thead>
<tr>
<th>img</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>image mode=RGB size=500x333 at 0x7F56F6360ED0&gt;</td>
<td>samoyed</td>
</tr>
<tr>
<td>image mode=RGB size=443x435 at 0x7F56F6379BD0&gt;</td>
<td>shiba_inu</td>
</tr>
<tr>
<td>image mode=RGB size=600x437 at 0x7F56F6379E50&gt;</td>
<td>Egyptian_Mau</td>
</tr>
<tr>
<td>image mode=RGB size=375x500 at 0x7F56F6379F90&gt;</td>
<td>Birman</td>
</tr>
<tr>
<td>image mode=RGB size=500x375 at 0x7F56F637B350&gt;</td>
<td>great_pyrenees</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>image mode=RGB size=500x335 at 0x7F56F623BD50&gt;</td>
<td>Bengal</td>
</tr>
<tr>
<td>image mode=RGB size=403x500 at 0x7F56F623BF90&gt;</td>
<td>leonberger</td>
</tr>
<tr>
<td>image mode=RGB size=500x375 at 0x7F56F6241390&gt;</td>
<td>beagle</td>
</tr>
<tr>
<td>image mode=RGB size=192x288 at 0x7F56F62416D0&gt;</td>
<td>Abyssinian</td>
</tr>
<tr>
<td>image mode=RGB size=288x300 at 0x7F56F6241910&gt;</td>
<td>american_pit_bull_terrier</td>
</tr>
</tbody>
</table>
Ask questions about your data (and models)

**Tables – DataFrames with rich media support**

❌ Classic DataFrames

<table>
<thead>
<tr>
<th>img</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>image mode=RGB size=500x333 at 0x7F56F6360ED0&gt;</td>
<td>samoyed</td>
</tr>
<tr>
<td>image mode=RGB size=443x435 at 0x7F56F6379BD0&gt;</td>
<td>shiba_inu</td>
</tr>
<tr>
<td>image mode=RGB size=600x437 at 0x7F56F6379E50&gt;</td>
<td>Egyptian_Mau</td>
</tr>
<tr>
<td>image mode=RGB size=375x500 at 0x7F56F6379F90&gt;</td>
<td>Birman</td>
</tr>
<tr>
<td>image mode=RGB size=500x375 at 0x7F56F637B350&gt;</td>
<td>great_pyrenees</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>image mode=RGB size=500x335 at 0x7F56F623BD50&gt;</td>
<td>Bengal</td>
</tr>
<tr>
<td>image mode=RGB size=403x500 at 0x7F56F623BF90&gt;</td>
<td>leonberger</td>
</tr>
<tr>
<td>image mode=RGB size=500x375 at 0x7F56F6241390&gt;</td>
<td>beagle</td>
</tr>
<tr>
<td>image mode=RGB size=192x288 at 0x7F56F62416D0&gt;</td>
<td>Abyssinian</td>
</tr>
<tr>
<td>image mode=RGB size=288x300 at 0x7F56F6241910&gt;</td>
<td>american_pit_bull_terrier</td>
</tr>
</tbody>
</table>

✅ wandb.Table

<table>
<thead>
<tr>
<th>img</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Image" /></td>
<td>samoyed</td>
</tr>
<tr>
<td><img src="image2" alt="Image" /></td>
<td>shiba_inu</td>
</tr>
<tr>
<td><img src="image3" alt="Image" /></td>
<td>Egyptian_Mau</td>
</tr>
<tr>
<td><img src="image4" alt="Image" /></td>
<td>Birman</td>
</tr>
</tbody>
</table>
Ask questions about your data (and models)

**Validated**  
**wandb.Table**

<table>
<thead>
<tr>
<th>img</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.jpg" alt="img1" /></td>
<td>samoyed</td>
</tr>
<tr>
<td><img src="image2.jpg" alt="img2" /></td>
<td>shiba_inu</td>
</tr>
<tr>
<td><img src="image3.jpg" alt="img3" /></td>
<td>Egyptian_Mau</td>
</tr>
<tr>
<td><img src="image4.jpg" alt="img4" /></td>
<td>Birman</td>
</tr>
</tbody>
</table>

```python
Table.groupby("label")
```

<table>
<thead>
<tr>
<th>Group by (label)</th>
<th>img</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.jpg" alt="img1" /></td>
<td><img src="image2.jpg" alt="img2" /> [:10 of 24]</td>
</tr>
<tr>
<td><img src="image3.jpg" alt="img3" /></td>
<td><img src="image4.jpg" alt="img4" /> [:5 of 34]</td>
</tr>
<tr>
<td><img src="image5.jpg" alt="img5" /></td>
<td><img src="image6.jpg" alt="img6" /> [:5 of 30]</td>
</tr>
<tr>
<td><img src="image7.jpg" alt="img7" /></td>
<td><img src="image8.jpg" alt="img8" /> [:5 of 25]</td>
</tr>
</tbody>
</table>
How do I ask questions about training?
Easily and systematically search hyperparameters
Sweeps – Easily search hyperparameters

Ask questions about your data (and models)

program: train.py
method: bayes
metric:
  name: valid_loss
  goal: minimize
parameters:
  batch_size:
    values: [32, 64]
  mixup_alpha:
    values: [0.2, 0.5, 0.8]
  optimizer:
    values: ["adam", "ranger"]
  encoder:
    values: ["resnet18", "resnet34", "resnet50", "resnet101"]
Three principles of an ideal ML workflow

- **Rapidly iterate**
  to continuously refine and optimize models

- **Reproduce**
  to reduce key-person dependencies

- **Collaborate**
  to ensure knowledge transfer across the organization
Example Pipeline

Raw Data → preprocess.py → Training Data

Pretrained Model → train.py → Fine-tuned Model

Eval Data → eval.py

Artifacts – Version data and models across your pipeline
Bad predictions!

Example Pipeline

Raw Data → preprocess.py

train.py → Fine-tuned Model

Pretrained Model

eval.py → Eval Data

Prediction: paper towel
Probability: 0.9871

Bad Pred eval.py
Bad predictions because of training?
Bad predictions because of **pretrained models**?

**Example Pipeline**

- **Raw Data**
- **preprocess.py**
- **Training Data**
  - **train.py**
  - **Fine-tuned Model**
  - **Bad Pred eval.py**
- **Eval Data**
- **Pretrained Model**
Artifacts – Version data and models across your pipeline

Bad predictions because of **preprocessing**?

Example Pipeline

- **Raw Data**
  - preprocess.py
  - Training Data
    - train.py
      - Pretrained Model
      - Fine-tuned Model
      - Eval Data
    - Eval Data
  - Bad Pred
    - eval.py
Artifacts – Version data and models across your pipeline

Bad predictions because of **raw data?**

Example Pipeline

- **Raw Data**
- preprocess.py
- **Training Data**
- train.py
- Fine-tuned Model
- Eval Data
- **Bad Pred**
- eval.py

- Pretrained Model
Bad predictions because of **multiprocessing**?
Bad predictions because of multiprocessing?
How do you debug a model pipeline?
Check the Code
Example Pipeline

**Artifacts** – Version data and models across your pipeline

**Problem with code? More than** `git diff`

Raw Data → preprocess.py → Training Data → train.py

- Raw Data
- preprocess.py
- Training Data
- train.py
- Fine-tuned Model
- Eval Data
- Bad Pred

`expand 22 lines...`

```
def read(self, frame):
    img = frame_to_ndarray(format="bgr24")
+    img = img[:1, 1, :1]  # convert bgr to rgb
    results = self.infer_img(img)
    processed_img = self.img_from_results(results)
    if random.random() > 0.9 and wandb.run is not None:
        self.wandb_log(img, processed_img)
-    return av.VideoFrame.from_ndarray(processed_img, format="bgr24")
+    return av.VideoFrame.from_ndarray(processed_img, format="rgb24")
```
Artifacts – Version data and models across your pipeline

What about the input/output artifacts?

Example Pipeline

- Raw Data → preprocess.py → Training Data → train.py → Fine-tuned Model
  - Pretrained Model → Eval Data
  - Bad Pred

Training Data
- Raw Data
- Eval Data
- Fine-tuned Model
Check the inputs
Bad predictions because of **Raw Data**

Example Pipeline

1. **Raw Data**
2. preprocess.py
3. Training Data
4. train.py
5. Fine-tuned Model
6. eval.py

**Artifacts** – Version data and models across your pipeline

Version: 1.0.3
Changed: March 12, 2022
By: Andrew Truong

**Diff:**
- Changed bounding box definitions
Good predictions after reverting Raw Data

Example Pipeline

Raw Data → preprocess.py → Training Data

Preprocess

Training Data → train.py → Fine-tuned Model

Pretrained Model

Fine-tuned Model → eval.py → Eval Data

Artifacts – Version data and models across your pipeline

Version: 1.0.2
Changed: March 11, 2022
By: Ben Sherman

Diff:
- Added 1713 new images
- Updated annotations.json

Prediction: redbone
Probability: 0.8861
Check the outputs
Tables – DataFrames with rich media support
Three principles of an ideal ML workflow

**Rapidly iterate**
- to continuously refine and optimize models

**Reproduce**
- to reduce key-person dependencies

**Collaborate**
- to ensure knowledge transfer across the organization
Interactive Dashboards

Check any metrics

Gourab 6:26 PM
Cool report, what about policy loss and CPU utilization?

Andrew Truong 6:27 PM
I didn’t think it was important — let me get back to you later...

Gourab 6:28 PM
It seems you don’t think anything I suggest as important.
Interactive Dashboards

Check any metrics

Gourab  6:26 PM
Cool report, what about policy loss and CPU utilization?

Andrew Truong  6:27 PM
I didn’t think it was important — I can take care of it...
Unified Reporting and Dashboarding
Open Source Research
The world’s leading ML teams trust us
The top open source research orgs use us:

- EleutherAI
- LAION
- CarperAI
- Harmonai
- OpenFold
- Craiyon
Integrations
Fits into your **workflow**

- Integrated into every popular ML framework
- Instrumented into over 6,000 popular ML repos
- Runs on every cloud or in your own infra
Competition Time!

wandb.me/xx
The W&B Course

www.wandb.courses
Thank You!
Join us and our ML community!

- Fully Connected – wandb.me/fc
- YouTube – wandb.me/youtube
- Twitter – wandb.me/twitter
Appendix
Andrew’s Presentation

https://www.youtube.com/watch?v=Se1HvbAM0O4&t=12s