

```
In [11]: data.show_batch(rows=3, figsize=(7,6))
```

miniature_pinscher



leonberger



japanese_chin



pug



havanese



```
In [13]: learn = cnn_learner(data, models.resnet34, metrics=error_rate)
```

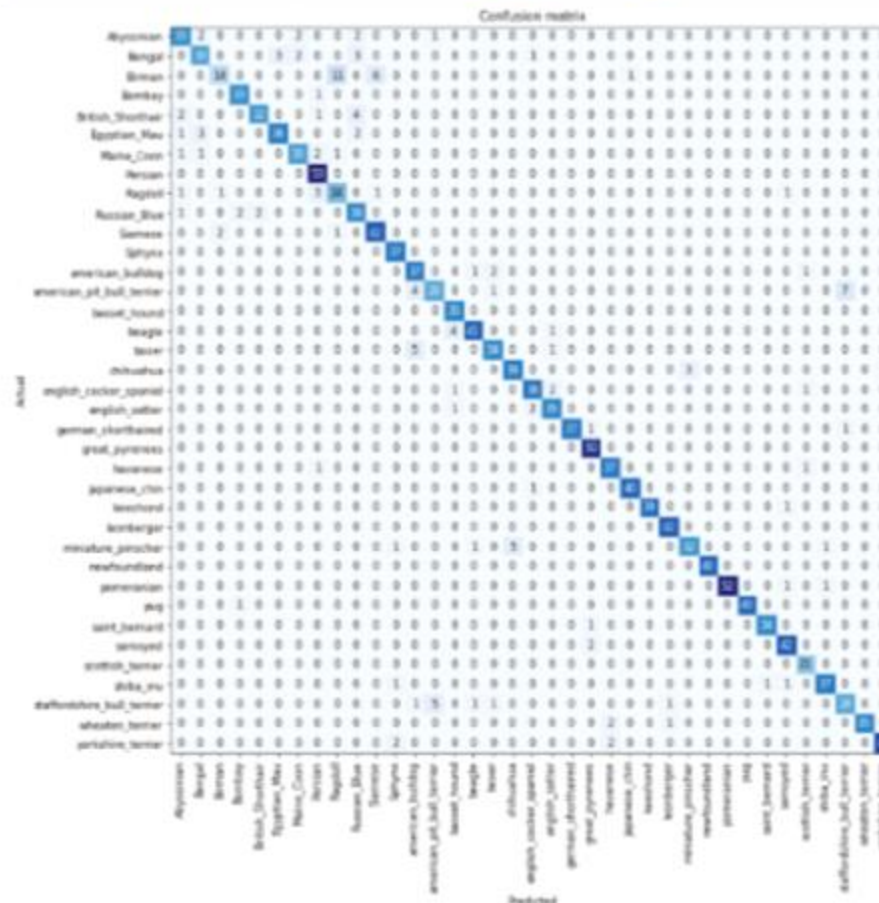
```
In [14]: learn.model
```

```
Out[14]: Sequential(
  (0): Sequential(
    (0): Conv2d(3, 3, bias=False)
    (1): BatchNorm1d(3)
    (2): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, ceil_mode=False)
    (4): Sequential(
      (0): BasicConv2d(64, 64, kernel_size=3, stride=1, padding=1, bias=True)
      adding=(1, 1), bn1: BatchNorm1d(64)
      track_running_stats=True
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=3, stride=1, padding=1, bias=True)
      adding=(1, 1), bn2: BatchNorm1d(64)
    )
  )
)
```

```
In [15]: learn.fit_one_cycle(4, 0.001)
```

epoch	train_loss	valid_loss
0	0.811533	0.100000
1	0.551540	0.100000
2	0.422648	0.100000
3	0.328013	0.100000

```
In [20]: interp.plot_confusion_matrix(figsize=(12,12), dpi=60)
```



fast.ai

Making neural nets
uncool again



Josiah Laivins
February 23, 2020

Making Deep Learning Accessible

Software

To make these available to use quickly, reliably, and with minimal code

Research

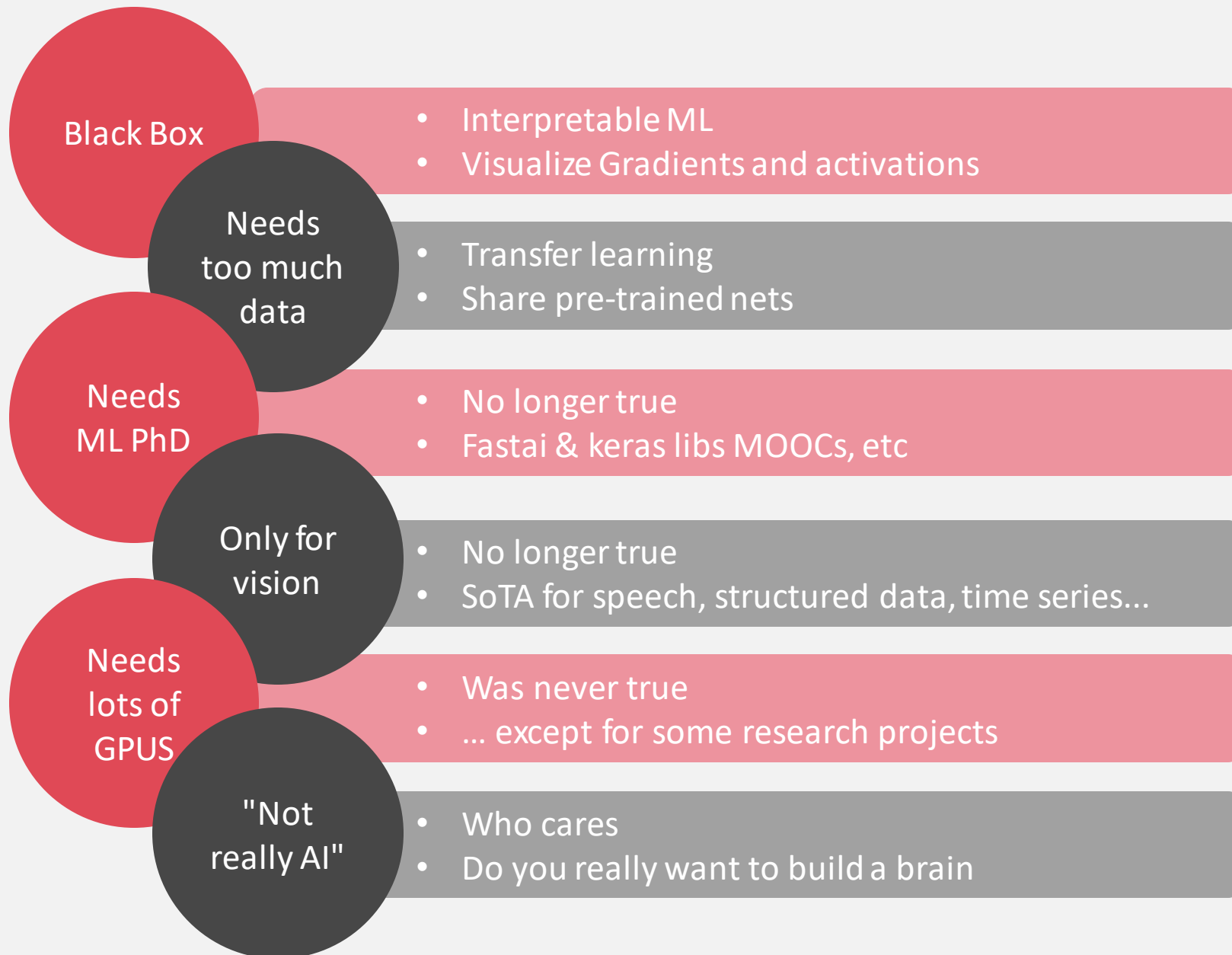
Ways to make state of the art deep learning techniques more accessible

Community

So that we can all help each other

Research

So that as many people as possible use these



fast.ai

Vision models

Computer Vision models zoo

The fastai library includes several pretrained models from [torchvision](#), namely:

- resnet18, resnet34, resnet50, resnet101, resnet152
- squeezenet1_0, squeezenet1_1
- densenet121, densenet169, densenet201, densenet161
- vgg16_bn, vgg19_bn
- alexnet

On top of the models offered by torchvision, fastai has implementations for the following models:

- Darknet architecture, which is the base of [Yolo v3](#)
- Unet architecture based on a pretrained model. The original unet is described [here](#), the model implementation is detailed in [models.unet](#)
- Wide resnets architectures, as introduced in [this article](#)

Darknet implimentation example

```
1 from ...torch_core import *
2 from ...layers import *
3
4 __all__ = ['Darknet', 'ResLayer']
5
6 def conv_bn_lrelu(ni:int, nf:int, ks:int=3, stride:int=1)->nn.Sequential:
7     "Create a seueue Conv2d->BatchNorm2d->LeakyRelu layer."
8     return nn.Sequential(
9         nn.Conv2d(ni, nf, kernel_size=ks, bias=False, stride=stride, padding=ks//2),
10        nn.BatchNorm2d(nf),
11        nn.LeakyReLU(negative_slope=0.1, inplace=True))
12
13 class ResLayer(Module):
14     "Resnet style layer with 'ni' inputs."
15     def __init__(self, ni:int):
16         self.conv1 = conv_bn_lrelu(ni, ni//2, ks=1)
17         self.conv2 = conv_bn_lrelu(ni//2, ni, ks=3)
18
19     def forward(self, x): return x + self.conv2(self.conv1(x))
20
21 class Darknet(Module):
22     "https://github.com/pjreddie/darknet"
23     def make_group_layer(self, ch_in:int, num_blocks:int, stride:int=1):
24         "starts with conv layer - 'ch_in' channels in - then has 'num_blocks' 'ResLayer'"
25         return [conv_bn_lrelu(ch_in, ch_in*2, stride=stride)
26                ] + [(ResLayer(ch_in*2)) for i in range(num_blocks)]
27
28     def __init__(self, num_blocks:Collection[int], num_classes:int, nf=32):
29         "create darknet with 'nf' and 'num_blocks' layers"
30         layers = [conv_bn_lrelu(3, nf, ks=3, stride=1)]
31         for i,nb in enumerate(num_blocks):
32             layers += self.make_group_layer(nf, nb, stride=2)
33             nf *= 2
34         layers += [nn.AdaptiveAvgPool2d(1), Flatten(), nn.Linear(nf, num_classes)]
35         self.layers = nn.Sequential(*layers)
36
37     def forward(self, x): return self.layers(x)
```

I wanna make my own model!

Trainer

If your model involves a novel way of training.

Model

If your model is simply a pytorch Module.

Learner

Needed if you have a new model and new Training method.

DataBunch

If there is a new way of handling data.

[1]: <https://github.com/fastai/fastai/blob/master/fastai/vision/gan.py>

[2]: <https://docs.fast.ai/vision.gan.html>

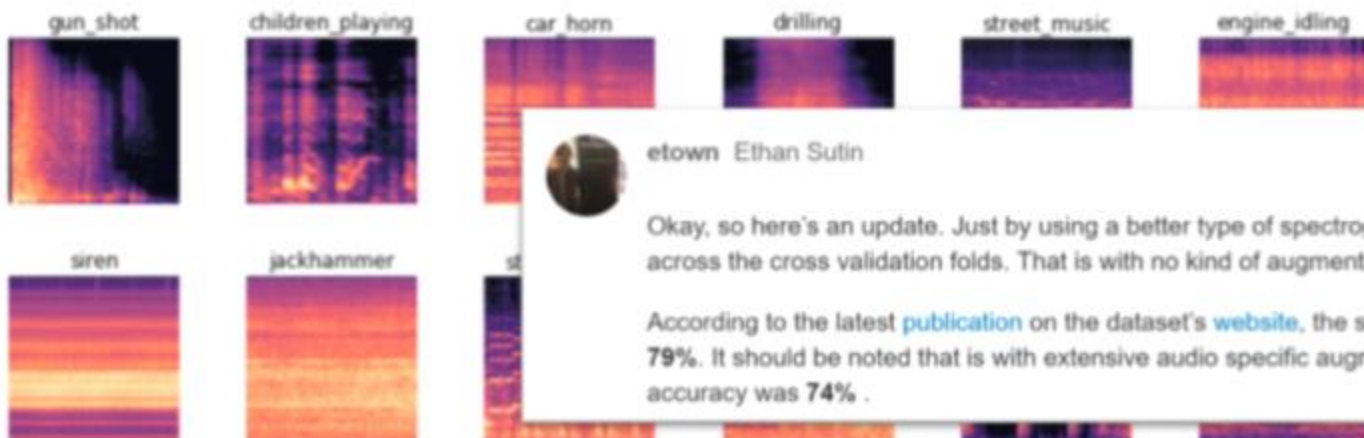
[3]: https://github.com/josiahls/fast-reinforcement-learning/blob/master/fast_rl/agents/dqn.py

Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification

Justin Salamon and Juan Pablo Bello

```
data = ImageDataBunch.from_folder(data_directory/'1', ds_tfms=[], size=224)
data.normalize(imagenet_stats)
```

```
data.show_batch(rows=6, figsize=(12,12))
```



etown Ethan Sutin



etown

1h

Okay, so here's an update. Just by using a better type of spectrogram, I was able to achieve **80.5%** accuracy across the cross validation folds. That is with no kind of augmentation at all.

According to the latest [publication](#) on the dataset's [website](#), the state-of-the-art mean accuracy achieved was **79%**. It should be noted that is with extensive audio specific augmentation, and without augmentation their top accuracy was **74%**.

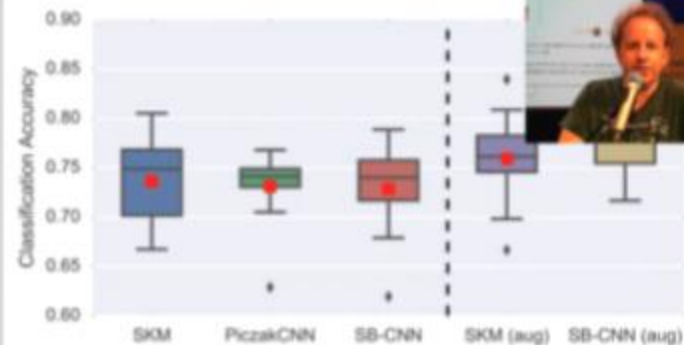
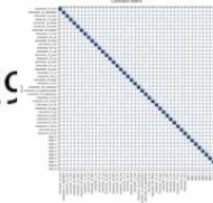


Fig. 1. Left of the dashed line: classification accuracy without augmentation – dictionary learning (SKM [7]), Piczak's CNN (PiczakCNN [11]) and the proposed model (SB-CNN). Right of the dashed line: classification accuracy for SKM and SB-CNN with augmentation.

[1]: <https://github.com/hiromis/notes/blob/master/Lesson2.md>

[2]: <https://forums.fast.ai/t/share-your-work-here/27676/215>

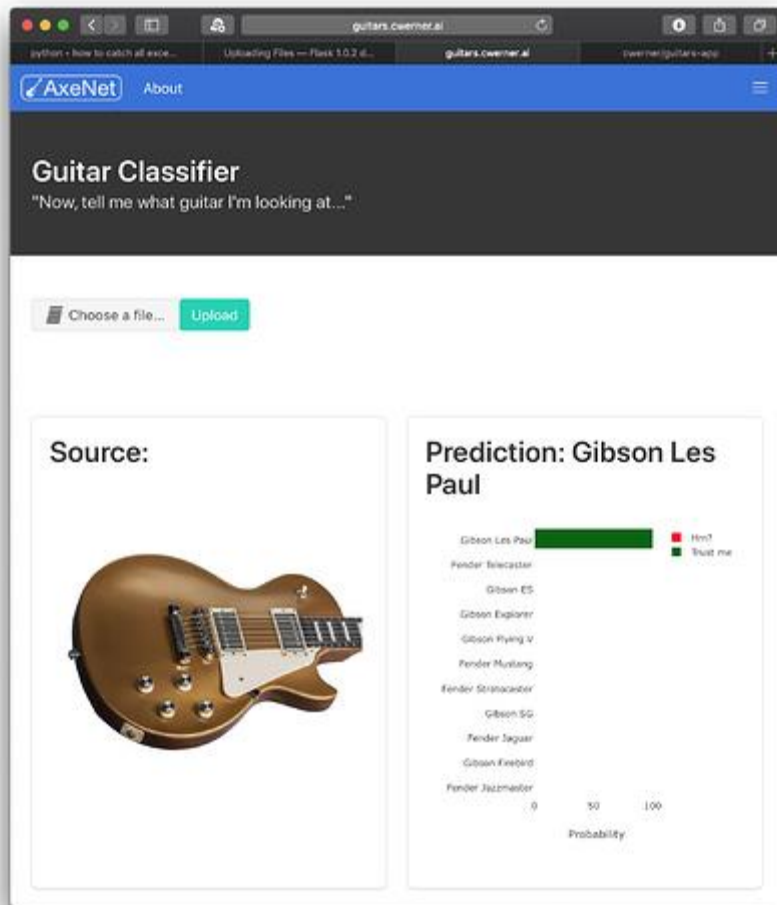
A man with curly hair, wearing a dark green t-shirt, is speaking into a microphone. He is standing behind a podium. In the background, there is a screen displaying a grid of Devanagari characters and a sign that says "DATA COMPLETE".



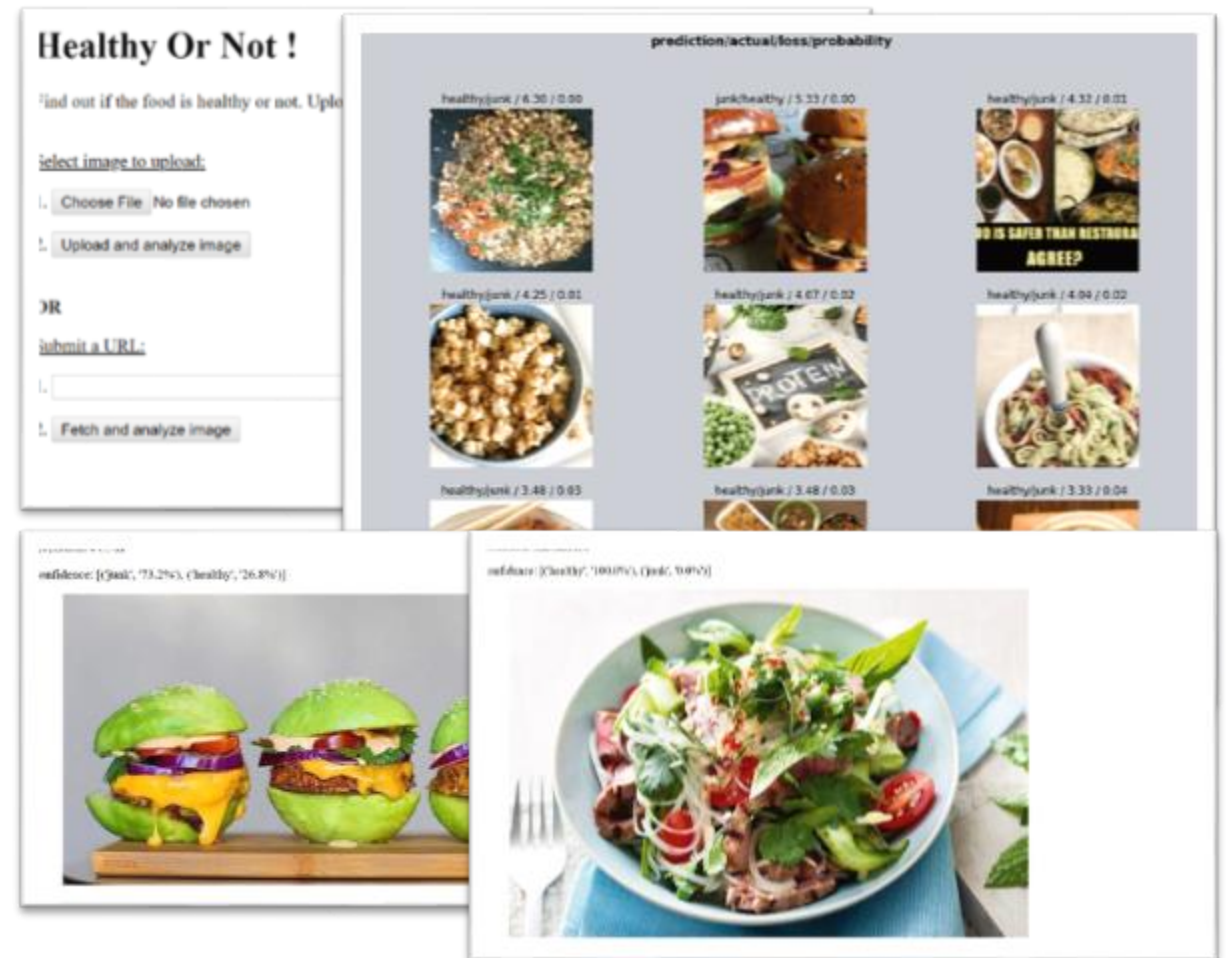
ter_24_dha/character_19_dha

- [1]: <https://github.com/hiromis/notes/blob/master/Lesson2.md>
[2]: <https://forums.fast.ai/t/share-your-work-here/27676/38>

Guitar Classifier



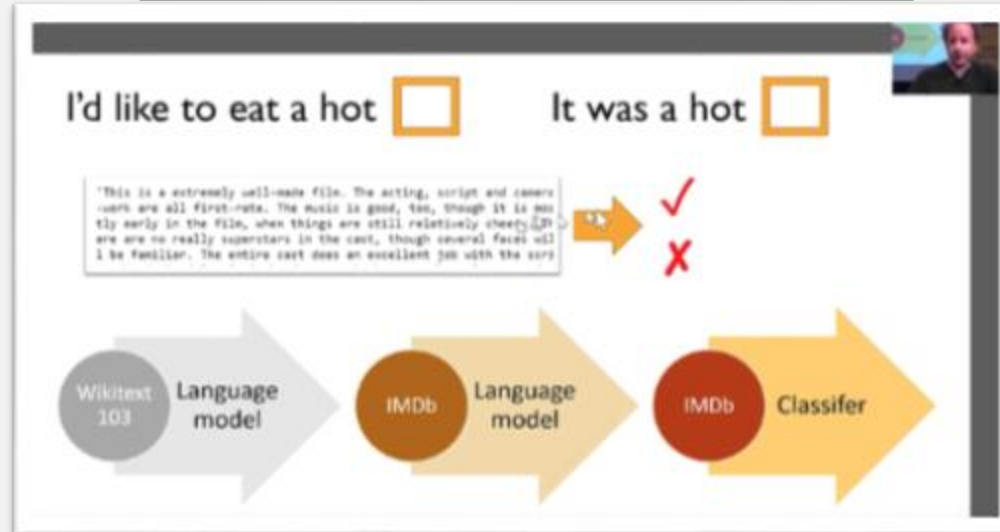
Junk or Health Food Classifier



- [1]: <https://github.com/hiromis/notes/blob/master/Lesson3.md>
[2]: <https://forums.fast.ai/t/share-your-work-here/27676/399>
[2]: <https://forums.fast.ai/t/share-your-work-here/27676/340>

fast.ai not just classification

Movie Review Sentiment Classification



Legal Document Classification

Legal Text Classifier with Universal Language Model Fine-tuning

LIM How Khang (LL.B. First Class Hon) (NUS), Advocate & Solicitor, Singapore

Overview

Recent work has shown that pre-training a neural language model on large public text datasets improves the accuracy of a neural text classifier while requiring fewer labelled training samples.

We use the Universal Language Model Fine-tuning method introduced by Howard and Ruder (2018)* to train a legal text classifier to perform 19-way legal topic classification on a dataset of 3,588 legal judgments issued by the Singapore Court of Appeal and High Court.

*Universal Language Model Fine-tuning for Text Classification
https://arxiv.org/pdf/1801.06439v1.pdf
https://github.com/harvardnlp/ud-pytorch

Dataset

The dataset was created by parsing the PDF Singapore Supreme Court judgments between 2000 and 2018 found on Singapore Law Watch (SLW), a free daily legal news website.

URL: <https://www.singaporelawwatch.sg/Judgments>

Size: 3,588 judgments (with only one fixed topic)

Topics: Civil Procedure, Criminal Law, Contract Law, Family Law, Companies, Tort, Arbitration, Legal Profession, Damages, Land, Insolvency Law, Building and Construction, Conflict of Laws, Administrative Law, Revenue Law, Trade Marks, Admiralty Law, Employment Law, Trusts.

Universal Language Model Fine-tuning by Howard and Ruder (2018)

Wikitext-103 Language Model

Specific Language Model

Specific Text Classifier

- Pre-trained language model on Wikitext-103
- Discriminative fine-tuning
- Started to regular learning rates
- Target task classifier fine-tuning with additional linear blocks and gradual unfreezing

Results

Accuracy: 82.56% accuracy on 19-way classification task

Time taken to fine-tune LM (GTX 1080 Ti): approx. 3h 40m

Time taken to train classifier (GTX 1080 Ti): approx. 1h

The model achieved strong results using the default settings in the fastai library. The incorrect test set predictions were reasonable errors due to an overlap in subject matter (e.g. Companies vs Insolvency Law, Revenue Law vs Land) and the generality of topics like Civil Procedure and Contract.

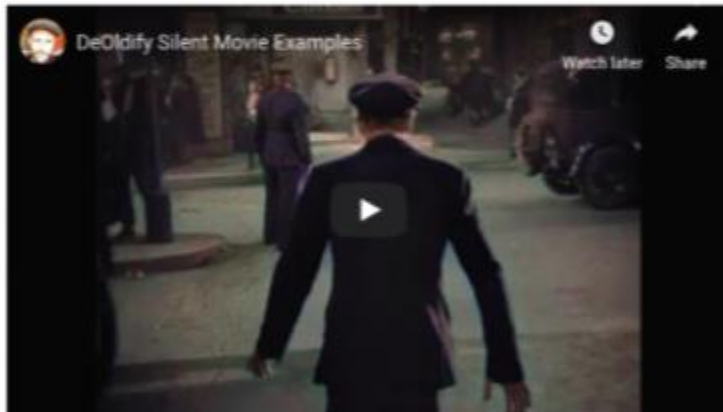
It would be useful to explore different levels of fine-tuning and whether multilabel classifier can achieve better results.

GANs

DeOldify

Decrappification, DeOldification, and Super Resolution

In this article we will introduce the idea of "decrappification", a deep learning method implemented in [fastai](#) on [PyTorch](#) that can do some pretty amazing things, like... colorize classic black and white movies—even ones from back in the days of silent movies, like this:



The same approach can make your old family photos look like they were taken on a modern camera, and even improve the clarity of microscopy images taken with state of the art equipment at the [Salk Institute](#), resulting in 300% more accurate cellular analysis.

Colorizer



[1]: <https://www.fast.ai/2019/05/03/decrappify/>

[2]: https://colab.research.google.com/github/jantic/DeOldify/blob/master/ImageColorizerColab.ipynb#scrollTo=LHfUPH42O_iK

Collaborative Filtering

Movie Recommendation System

"How's that movie?"- Neural collaborative filtering with FastAI

Build a state-of-the-art movie recommendation system with just 10 lines of code



Looking at some predictions

While it's great to see the loss go down, let's look at some actual predictions of the model.

```
(users, items), ratings = next(iter(data.valid_dl))
preds = learn.model(users, items)
print('Real\tPred\tDifference')
for p in list(zip(ratings, preds))[:16]:
    print('{}\t{}\t{}'.format(p[0], p[1], p[1]-p[0]))
```

Real	Pred	Difference
5.0	4.2	-0.8
4.0	3.2	-0.8
3.0	3.4	0.4
3.0	2.4	-0.6
3.0	2.9	-0.1
2.0	2.4	0.4
4.0	4.2	0.2
4.0	4.6	0.6
4.0	3.6	-0.4
1.0	2.3	1.3
5.0	4.3	-0.7
2.0	2.9	0.9
4.0	4.0	0.0
4.0	3.7	-0.3

Book Recommendation System

```
1 learn = collab_learner(data, n_factors=40, y_range=(1, 5), wd=1e-1)
```

Output:

```
Top idx:
array(['5000', '3315', '3313', '3312', '3311', '3309', '3308',
       '3307', '3306', '3304'], dtype='<U21')
```

Top names:

```
array(['Passion Unleashed (Demonica #3)', 'My Story', 'The Gargoyle',
       'Pretty Baby', ...,
       'Top Secret Twenty-One (Stephanie Plum, #21)', 'The Warrior
       Heir (The Heir Chronicles, #1)', 'Stone Soup',
       'The Sixth Man (Sean King & Michelle Maxwell, #5)'],
      dtype='<U144')
```

Most negative bias:

```
[(tensor(-0.1021), 'The Almost Moon', 2.49),
 (tensor(-0.0341), 'Skinny Bitch', 2.9),
 (tensor(-0.0325), 'Bergdorf Blondes', 3.0),
 (tensor(-0.0316), 'The Particular Sadness of Lemon Cake', 2.93),
 (tensor(-0.0148), 'The Weird Sisters', 3.08)]
```

[1]: <https://course.fast.ai/videos/?lesson=4>

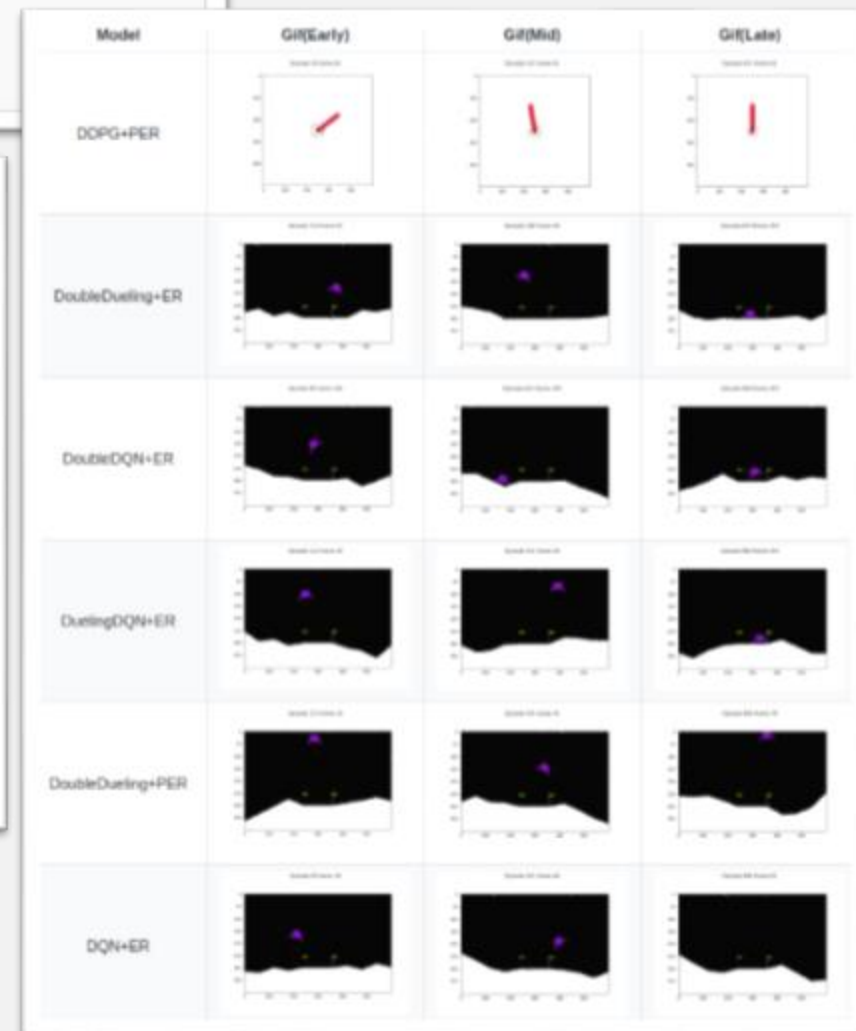
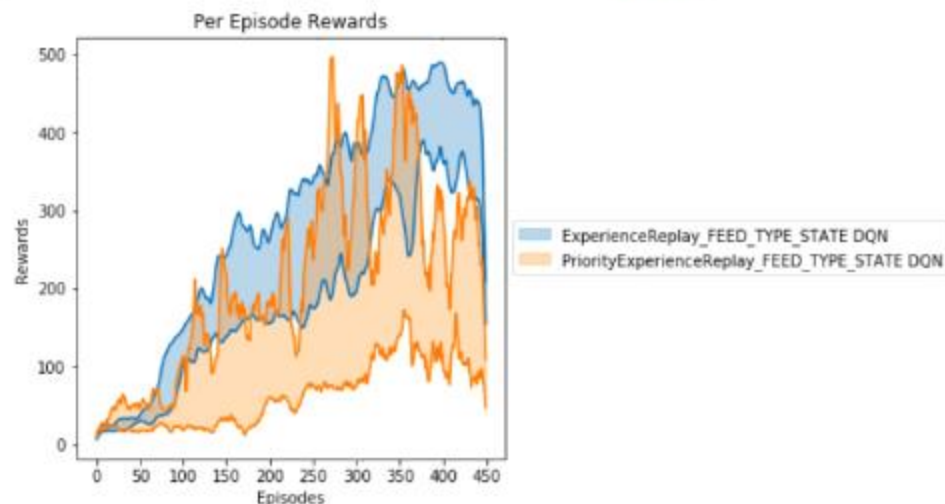
[2]: <https://towardsdatascience.com/collaborative-filtering-with-fastai-3dbdd4ef4f00>

[3]: <https://jovian.ml/aakashns/5bc23520933b4cc187cfe18e5dd7e2ed>

Reinforcement Learning (experimental)

```
In [ ]: memory = ExperienceReplay(memory_size=1000000, reduce_ram=True)
        explore = GreedyEpsilon(epsilon_start=1, epsilon_end=0.1, decay=0.001)
        model = create_dqn_model(data=data, base_arch=DQNModule, lr=0.001, layers=[32,32], opt=optim.RM
        Sprop)
        learn = dqn_learner(data, model, memory=memory, exploration_method=explore,
        callback_fns=[RewardMetric, EpsilonMetric])
        learn.fit(3)
```

```
In [16]: per_group_interp = GroupAgentInterpretation.from_pickle('data/cartpole_dqn', 'dqn_PriorityExper
        ienceReplay_FEED_TYPE_STATE')
        per_group_interp.add_interpretation(group_interp)
        per_group_interp.plot_reward_bounds(per_episode=True, smooth_groups=10)
```



fast.ai has a MOOC!

