fast.ai v3 part 2 PyTorch learning group

It is so cool to be here, I am Sebastian.

Today, we gonna take a look at...

Data Block API

Generic Optimizers

Augmentation

A practical application of NLP

It is going to be...

… a repetition - we gonna dive into the lecture

… practice related

… inspiring

All you need is practice!

“It works as a motivator to entice you to put in the work.”

Manuel Amunategui. “Monetizing Machine Learning”

THEORY

Layerwise Sequential Unit Variance (LSUV)

Target: have activations that stay normalized (i.e., mean 0, sd 1)

How: Let the computer figure it out!

1. Initialize neural net with the usual technique.
2. Pass a batch through the model.
3. Check the outputs of the linear and convolutional layers.
4. Subtract the mean from the initial bias.
5. Rescale the weights according to the actual variance on the activations.

```
def lsuv_module(m, xb):
    h = Hook(m, append_stat)
    (4) while mdl(xb) is not None and abs(h.mean)  > 1e-3: m.bias -= h.mean
    (5) while mdl(xb) is not None and abs(h.std-1) > 1e-3: m.weight.data /= h.std
    h.remove()
    return h.mean,h.std
```

All You Need is a Good Init

Data Block API

What did you guys like?

Things I liked in 08_data_block

Monkey-patching the ls function to look into a directory from a notebook:

```
Path.ls = lambda x: list(x.iterdir())
```

Use the MIME types database to ignore everything that is not an image:

```
image_extensions = set(k for k,v in mimetypes.types_map.items() if v.startswith('image/'))
```

Walk (potentially recursively) through all the folder in path:

```
def get_files(path, extensions=None, recurse=False, include=None):
    path = Path(path); extensions = setify(extensions); extensions = {e.lower() for e in extensions}
    if recurse:
        res = []; for i,(p,d,f) in enumerate(os.walk(path)): # returns (dirpath, dirnames, filenames)
            if include is not None and i==0: d[:] = [o for o in d if o in include]
            else:                            d[:] = [o for o in d if not o.startswith('.')]
            res += _get_files(p, f, extensions)
        return res
    else:
        f = [o.name for o in os.scandir(path) if o.is_file()]
        return _get_files(path, f, extensions)
```

Prepare for modeling

What we need to do:

1. Get files
2. Split validation set by random%, folder name, csv, ...
3. Label by folder name, file name/re, csv, ...
4. Transform per image (optional)
5. Transform to tensor
6. DataLoader
7. Transform per batch (optional)
8. DataBunch
9. Add test set (optional)

https://colab.research.google.com/drive/1XUKOkp73bYWnt_OQN25HFLOHP8ovvcy2

Optimizers

How would you use them in NLP?

Intro to optimizers

Pytorch approach:

- Base optimizer is dictionary that stores hyper-parameters and references to the parameters
- contains a method **step** that will update parameters with the gradients and a method zero_grad to detach and zero the gradients of all parameters

FastAI approach:

- Equivalent from scratch, but more flexible
- step function loops over all the parameters to execute the step using stepper functions that need to be provided when initializing the optimizer.

```
class Optimizer():
    def __init__(self, params, steppers, **defaults):
        # might be a generator
        self.param_groups = list(params)
        # ensure params is a list of lists
        if not isinstance(self.param_groups[0], list): self.param_groups = 
[self.param_groups]
        self.hypers = [{**defaults} for p in self.param_groups]
        self.steppers = listify(steppers)

    def grad_params(self):
        return [(p,hyper) for pg,hyper in zip(self.param_groups,self.hypers)
            for p in pg if p.grad is not None]

    def zero_grad(self):
        for p,hyper in self.grad_params():
            p.grad.detach_()
            p.grad.zero_()

    def step(self):
        for p,hyper in self.grad_params(): compose(p, self.steppers, **hyper)
```

WD with Momentum

Momentum requires to add some state. We need to save the moving average of the gradients to be able to do the step and store this inside the optimizer state. To do this, we introduce statistics. Statistics are object with two methods:

- init_state, that returns the initial state (a tensor of 0. for the moving average of gradients)
- update, that updates the state with the new gradient value

We also read the _defaults values of those objects, to allow them to provide default values to hyper-parameters.

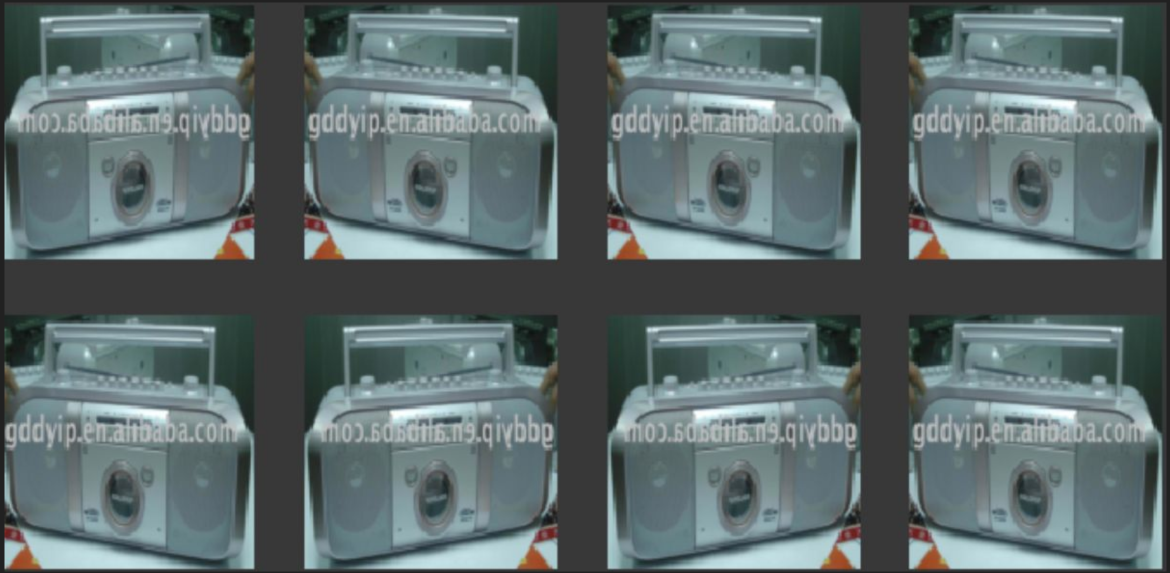
```
class StatefulOptimizer (Optimizer):
    def __init__(self, params, steppers, stats=None,
**defaults):
        self.stats = listify(stats)
        maybe_update (self.stats, defaults, get_defaults)
        super().__init__(params, steppers, **defaults)
        self.state = {}

    def step(self):
        for p,hyper in self.grad_params():
            if p not in self.state:
                #Create a state for p and call all the
statistics to initialize it.
                self.state[p] = {}
                maybe_update (self.stats, self.state[p],
lambda o: o.init_state(p))
            state =  self.state[p]
            for stat in self.stats: state = stat.update (p,
state, **hyper)
            compose (p, self.steppers, **state, **hyper)
            self.state[p] = state
```

Adam and friends...

Augmentation

PIL transformations - random flip

```
def pil_random_flip(x):
    return x.transpose(PIL.Image.FLIP_LEFT_RIGHT) if random.random()<0.5 else x
```

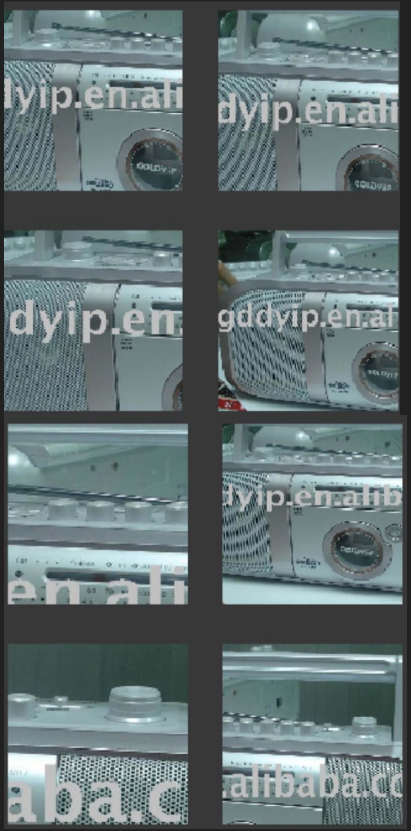


PIL transformations - random crop

Time budget calculations:

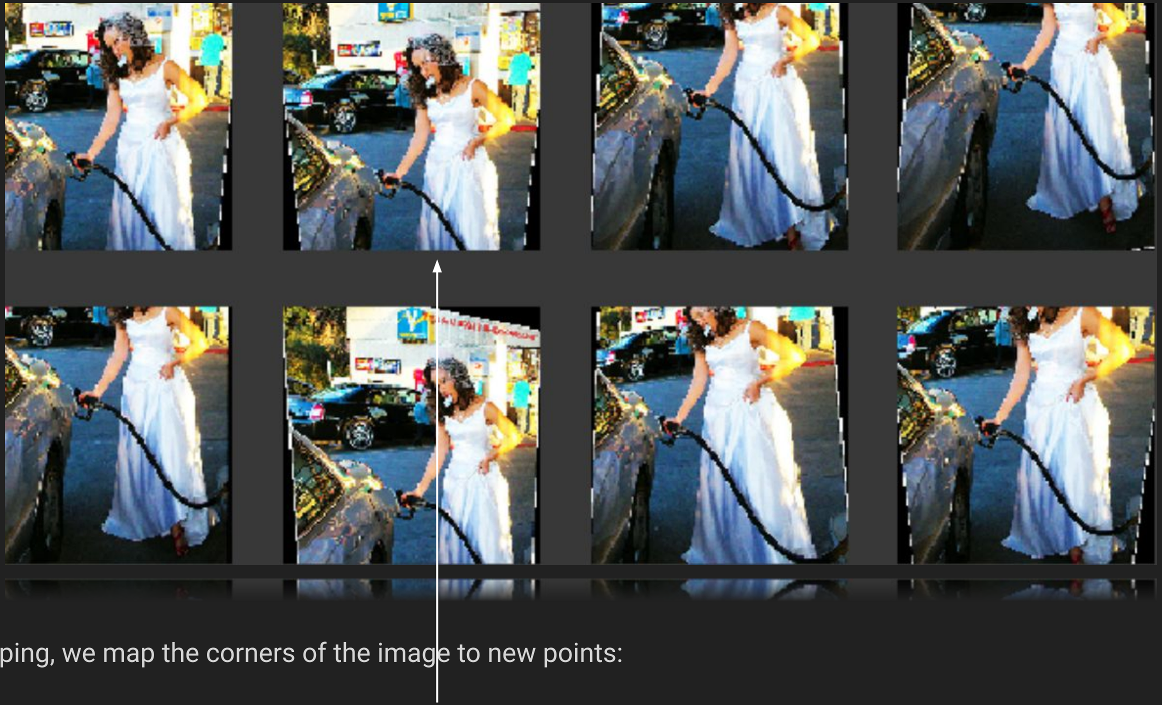
- Time budget: aim for 5 mins per batch for imagenet on 8 GPUs.
- 1.25m images in imagenet.
- On one GPU per minute that's `1250000/8/5 == 31250`, or 520 per second.
- Assuming 4 cores per GPU, then we want ~125 images per second.
- So try to stay <10ms per image.

PIL transformations - random resize crop



```
class RandomResizedCrop (GeneralCrop):
    def __init__(self, size, scale=(0.08,1.0), ratio=(3./4., 4./3.), resample=PIL.Image.BILINEAR) :
        super ().__init__(size, resample=resample )
        self.scale,self.ratio = scale ,ratio

    def get_corners (self, w, h, wc, hc):
        area = w*h
        #Tries 10 times to get a proper crop inside the image.
        for attempt in range(10):
            area = random.uniform (*self.scale) * area
            ratio = math.exp (random.uniform (math.log (self.ratio[0]), math.log (self.ratio[1])))
            new_w =  int(round(math.sqrt (area * ratio)))
            new_h =  int(round(math.sqrt (area / ratio)))
            if new_w <= w and new_h <= h:
                left = random.randint (0, w - new_w)
                top  = random.randint (0, h - new_h)
                return (left, top, left + new_w, top + new_h)

        # Fallback to squish
        if   w/h < self.ratio[0]: size = (w, int(w/self.ratio[0]))
        elif w/h > self.ratio[1]: size = (int(h*self.ratio[1]), h)
        else:                     size = (w, h)
        return ((w-size[0])//2, (h-size[1])//2, (w+size[0])//2, (h+size[1])//2)
```

PIL transformations - perspective warping



To do perspective warping, we map the corners of the image to new points:

- To tilt the image so that the top looks closer to us => top/left corner needs to be shifted to the right and the top/right to the left
- To avoid squishing, the bottom/left corner needs to be shifted to the left and the bottom/right corner to the right.

Batch Data Augmentation

- Write your own augmentation for your domain's data types, and have them run on the GPU, by using regular PyTorch tensor operations.
  - The key is to do them on a whole batch at a time.
  - Nearly all PyTorch operations can be done batch-wise.
  - Once we have batched the data together, we can apply more data augmentation on a batch level.

- **What augmentations would you write in NLP?**

PRACTICE

What is the business guy doing here?

amorous.me - an NLP project

Thank you for your attention!