Image Captioning In PyTorch

March 21st, 2019
8:00 - 10:00 am
GPU Technology Conference (GTC)

Laura N Montoya

Executive Director, Accel AI Institute
Co-Chair LatinX in AI Coalition

NVIDIA

LXAI

Accel.AI

Laura N Montoya

I am a scientist, engineer, and social impact entrepreneur.

You can find me at...

@quickresolute

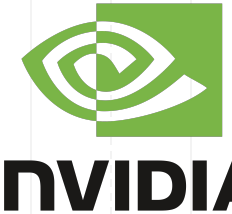
info@accel.ai

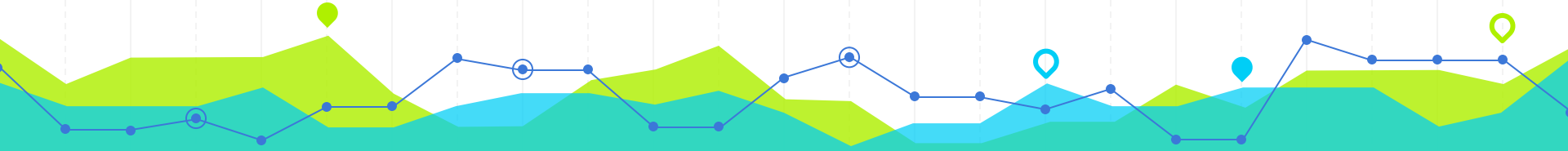
www.lauranmontoya.com

Accel.AI

LXAI

ACCEL IMPACT





Intro to Image Captioning in PYTORCH

DOCS & DOWNLOADS

Accel.AI

OFFICIAL DOCS

Python

https://docs.python.org/2.7/

Anaconda

https://www.anaconda.com/download/

PyTorch

https://pytorch.org/

Data

Coco Image Dataset

http://cocodataset.org/#download

GITHUB

Intro to Image Captioning

https://github.com/latinxinai/Intro-Image-Captioning-Lab

SCIENTIFIC PACKAGES

Python Modules

- Pathlib - Object-oriented filesystem paths
- Pandas - Data Analysis Library
- OS - operating system interface
- Numpy - n-dimensional array objects
- Matplotlib - object-oriented API for embedding plots into applications

ML & DL Modules

- OpenCV - computer vision library
- Spacy - Natural Language Processing

Accel.AI

AGENDA

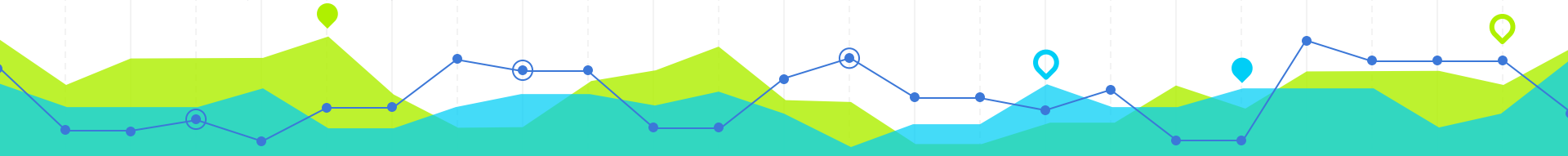
Accel.AI

Main Concepts

1. COCO Image Dataset
2. Image Data Augmentation
3. Build Vocabulary
4. Prepare Dataset
5. What models are used?
6. CNN (Encoder)
7. RNN (Decoder)
8. Training
9. Evaluation
10. Image Captioning Applications

Applied AI Labs

1. Intro to Pytorch
2. Image Captioning in PyTorch



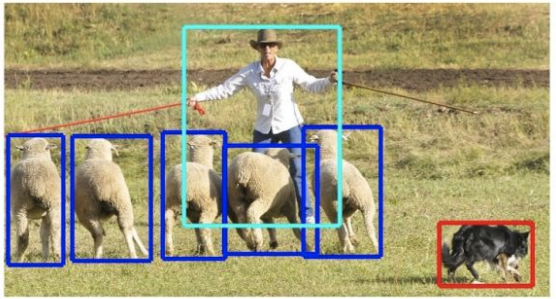
MS COCO Image Dataset

Common Objects in Context

1

Accel.AI

(a) Image classification



(b) Object localization

(c) Semantic segmentation

(d) This work

COCO

Common Objects in Context

[Microsoft COCO: Common Objects in Context]

Accel.AI

a pot of broccoli on a stove
a wok with a cooked broccoli meal in it

a herd of cattle standing on top of a grass covered field
several cows are gathered together in a grassy field

a yellow sign on a dirt road
a floodway sign sitting on the side of a road in a field

a group of people posing for a picture on a ski lift
three people wearing ski gear sitting on a ski lift

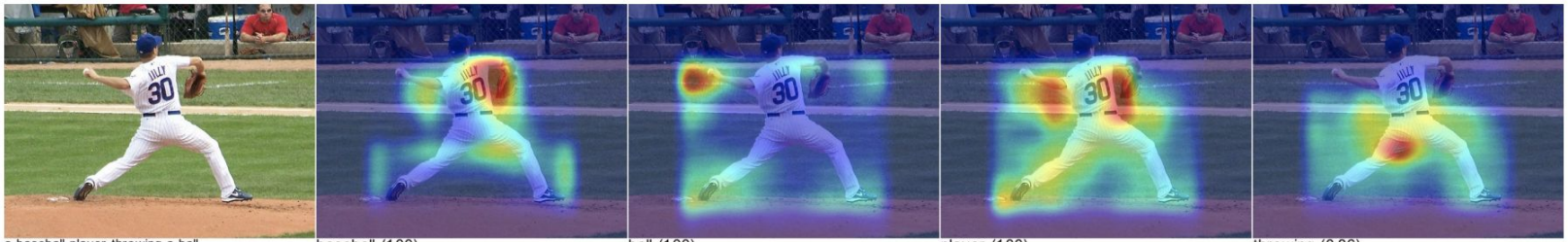
a man sitting on a couch with a dog
a man sitting on a chair with a dog in his lap

dog (1.00)

man (0.93)

sitting (0.83)

couch (0.66)



a baseball player throwing a ball
a pitcher holds his arm far behind him during a pitch

baseball (1.00)

ball (1.00)

player (1.00)

throwing (0.86)

Microsoft Research blog

Accel.AI

2014 Train/Val	Detection 2015, Captioning 2015, Detection 2016, Keypoints 2016
2014 Testing	Captioning 2015
2015 Testing	Detection 2015, Detection 2016, Keypoints 2016
2017 Train/Val/Test	Detection 2017, Keypoints 2017, Stuff 2017, Detection 2018, Keypoints 2018, Stuff 2018, Panoptic 2018
2017 Unlabeled	[optional data for any competition]

Throughout the API "ann"=annotation, "cat"=category, and "img"=image.

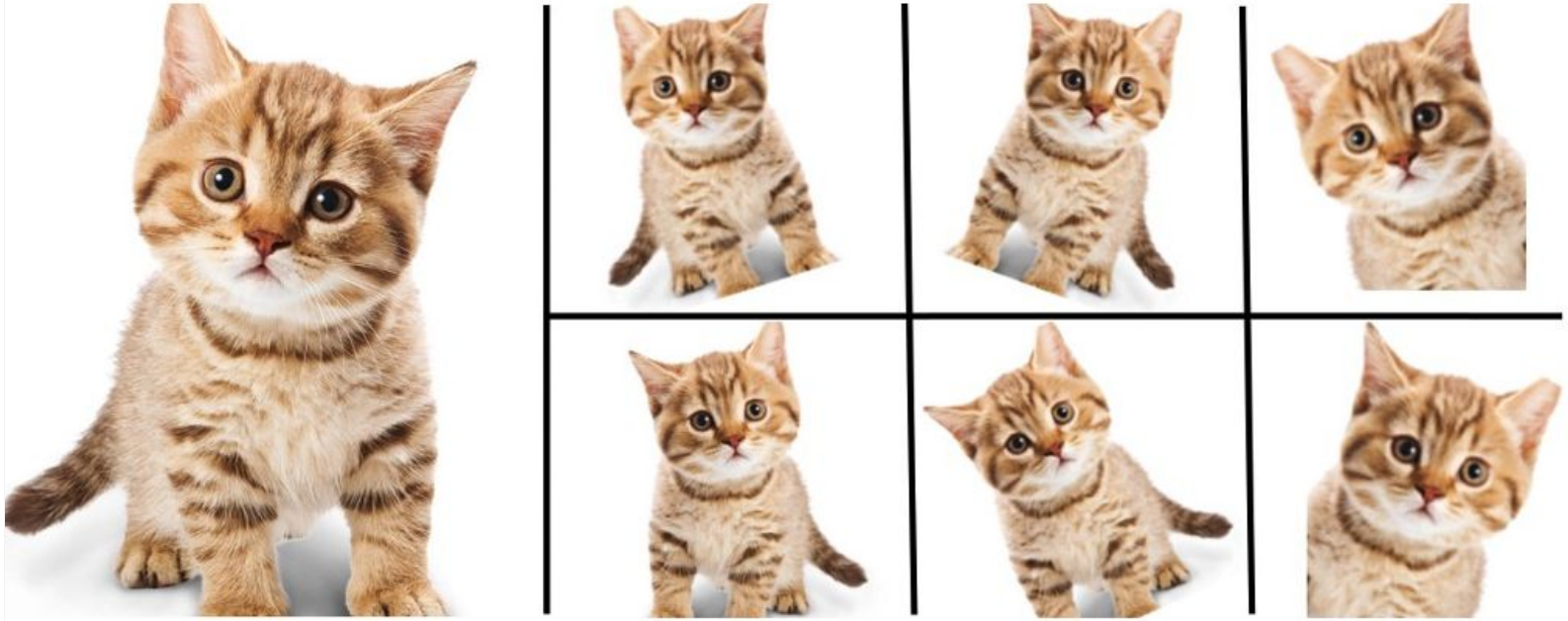
`getAnnIds`	Get ann ids that satisfy given filter conditions.
`getCatIds`	Get cat ids that satisfy given filter conditions.
`getImgIds`	Get img ids that satisfy given filter conditions.
`loadAnns`	Load anns with the specified ids.
`loadCats`	Load cats with the specified ids.
`loadImgs`	Load imgs with the specified ids.
`loadRes`	Load algorithm results and create API for accessing them.
`showAnns`	Display the specified annotations.

Data Augmentation

Optimizing Transformations

2

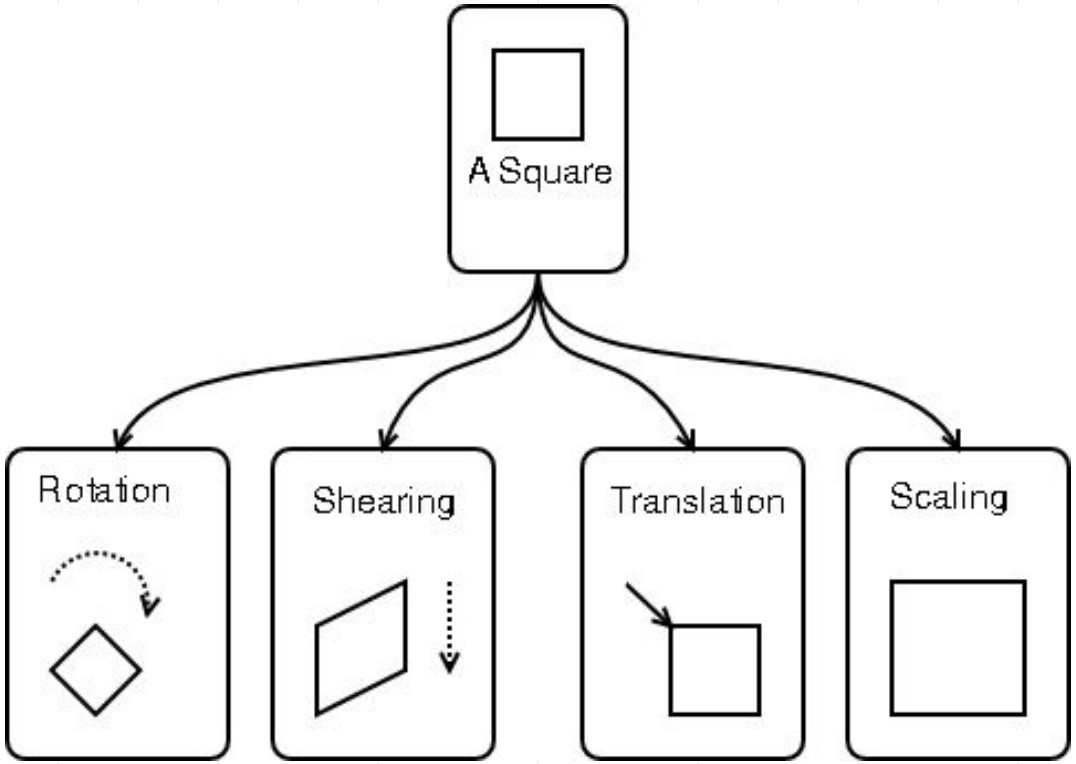
Accel.AI



Enlarge your Dataset

[Data Augmentation Blog, Bharath Raj]

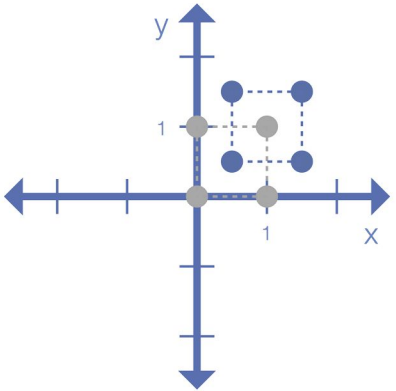
AFFINE TRANSFORMATIONS



[Libart Library: Affine Transformation Matrices]

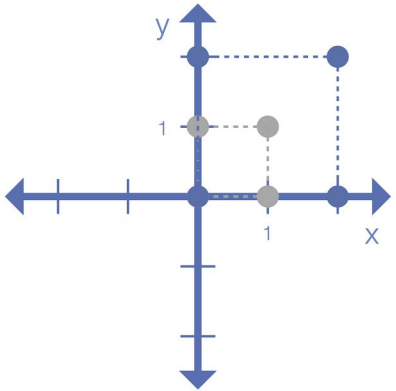
Translate

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$



Scale

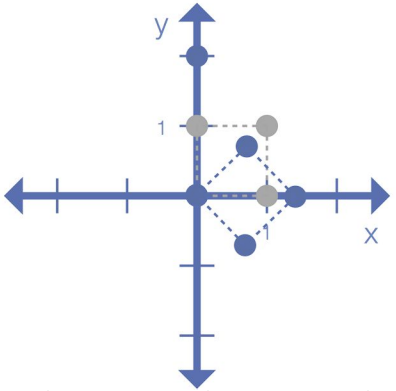
$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Rotate

$$\begin{bmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$c = s = \sin(45°)$



Shear

$$\begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

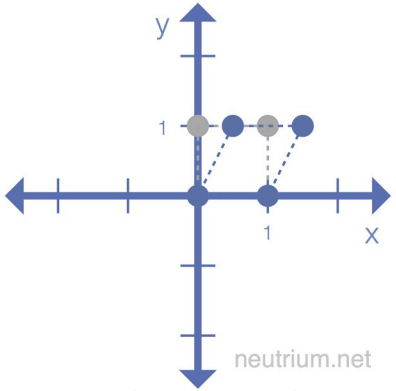
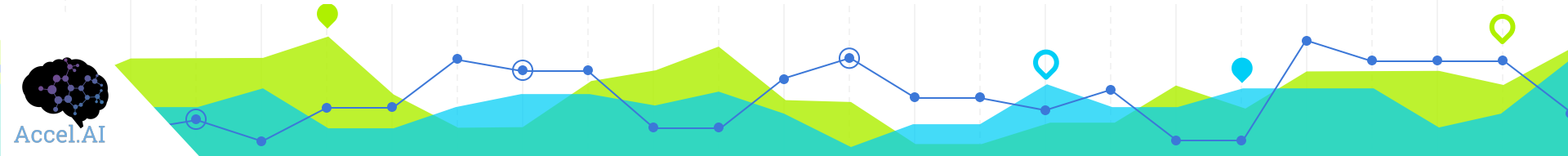


IMAGE CROPPING

```
import math
def center_crop(im, min_sz=None):
    """ Returns a center crop of an image"""
    r,c,*_ = im.shape
    if min_sz is None: min_sz = min(r,c)
    start_r = math.ceil((r-min_sz)/2)
    start_c = math.ceil((c-min_sz)/2)
    return crop(im, start_r, start_c, min_sz, min_sz)

def crop(im, r, c, target_r, target_c): return im[r:r+target_r, c:c+target_c]

def random_crop(x, target_r, target_c):
    """ Returns a random crop"""
    r,c,*_ = x.shape
    rand_r = random.uniform(0, 1)
    rand_c = random.uniform(0, 1)
    start_r = np.floor(rand_r*(r - target_r)).astype(int)
    start_c = np.floor(rand_c*(c - target_c)).astype(int)
    return crop(x, start_r, start_c, target_r, target_c)

def rotate_cv(im, deg, mode=cv2.BORDER_REFLECT, interpolation=cv2.INTER_AREA):
    """ Rotates an image by deg degrees"""
    r,c,*_ = im.shape
    M = cv2.getRotationMatrix2D((c/2,r/2),deg,1)
    return cv2.warpAffine(im,M,(c,r), borderMode=mode, flags=cv2.WARP_FILL_OUTLIERS+interpolation)
```

```
# center crop
im_crop = center_crop(im)
plt.imshow(im_crop)
```

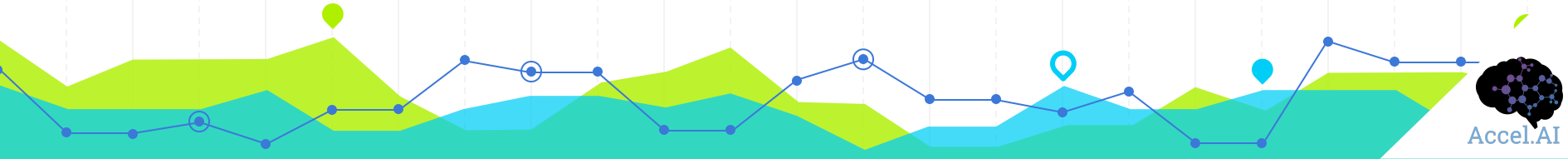


ROTATION & FLIPPING IMAGES

```
# Random Rotation (-10, 10)
rdeg = (np.random.random()-.50)*20
print(rdeg)
im_rot = rotate_cv(im, rdeg)
plt.imshow(im_rot)
```

```
3.5133900965477127
```

```
# Horizontal flip
im_f = np.fliplr(im)
plt.imshow(im_f)
```



Accel.AI

IMAGE RESIZING

```
def read_image(path):
    im = cv2.imread(str(path))
    return cv2.cvtColor(im, cv2.COLOR_BGR2RGB)

# resize and center crop
def resize_crop_image(path, sz=(250, 250)):
    im = read_image(path)
    im = center_crop(im)
    return cv2.resize(im, sz)
```

```
path = PATH/"train2014/COCO_train2014_000000356702.jpg"
im = resize_crop_image(path)
plt.imshow(im)
```

RESIZE ALL IMAGES

```
def resize_all_images(resize_path, org_path):
    files = [x for x in list(org_path.iterdir()) if x.suffix == ".jpg"]
    for f in files:
        f_name = f.parts[-1]
        new_path = resize_path/f_name

        im = resize_crop_image(f)
        cv2.imwrite(str(new_path), cv2.cvtColor(im, cv2.COLOR_RGB2BGR))
```

```
resize_train = PATH/"train_resized"
resize_train.mkdir(exist_ok=True)
resize_val = PATH/"val_resized"
resize_val.mkdir(exist_ok=True)
```

```
#resize_all_images(resize_train, PATH/"train2014")
```

```
#resize_all_images(resize_val, PATH/"val2014")
```

Build Vocabulary

Words to Vectors

3

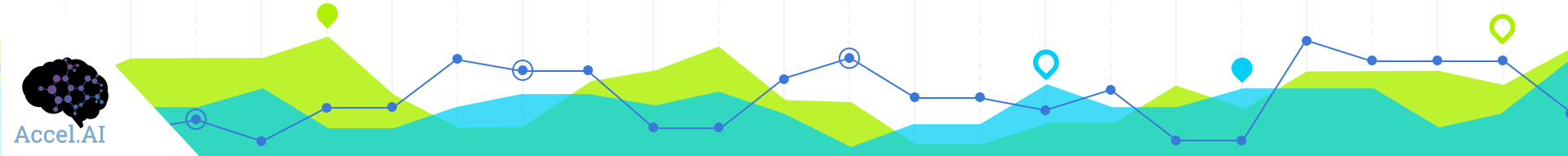
Accel.AI

TOKENIZATION WITH SPACY

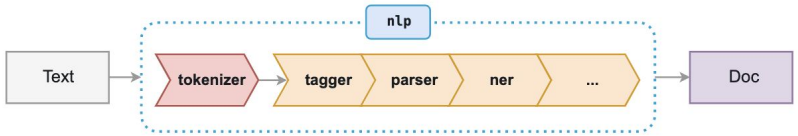
```
# tokenization
# first time run this
#!python3 -m spacy download en

my_tok = spacy.load('en')
def spacy_tok(x): return [tok.text for tok in my_tok.tokenizer(x)]
```

```
coco = COCO(PATH/"annotations/captions_train2014.json")
annIds = list(coco.anns.keys())
caption = coco.anns[annIds[0]]['caption']
spacy_tok(caption.lower())
```

```
loading annotations into memory...
Done (t=0.86s)
creating index...
index created!
['a', 'very', 'clean', 'and', 'well', 'decorated', 'empty', 'bathroom']
```

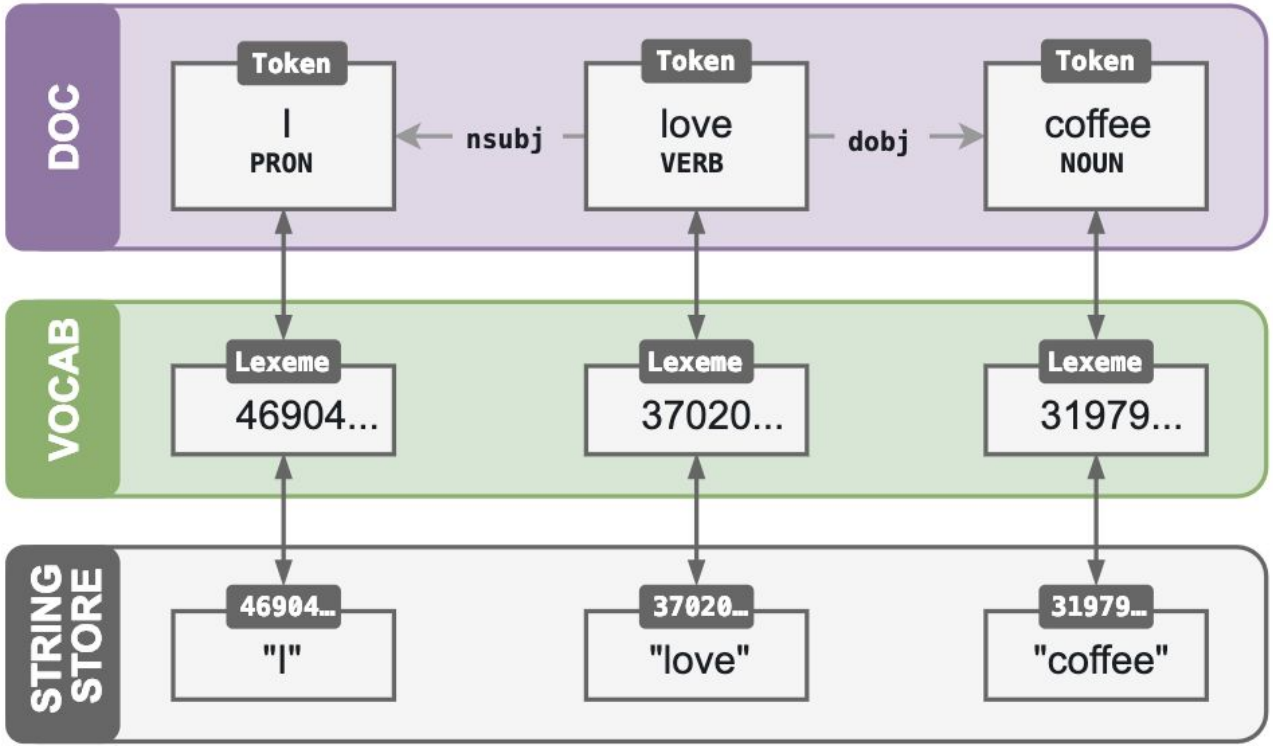


SPACY PIPELINES



NAME	COMPONENT	CREATES	DESCRIPTION
tokenizer	`Tokenizer`	`Doc`	Segment text into tokens.
tagger	`Tagger`	`Doc[i].tag`	Assign part-of-speech tags.
parser	`DependencyParser`	`Doc[i].head`, `Doc[i].dep`, `Doc.sents`, `Doc.noun_chunks`	Assign dependency labels.
ner	`EntityRecognizer`	`Doc.ents`, `Doc[i].ent_iob`, `Doc[i].ent_type`	Detect and label named entities.
textcat	`TextCategorizer`	`Doc.cats`	Assign document labels.
...	custom components	`Doc._.xxx`, `Token._.xxx`, `Span._.xxx`	Assign custom attributes, methods or properties.

SPACY VOCAB


DOC
Token
I
PRON
nsubj
Token
love
VERB
dobj
Token
coffee
NOUN
VOCAB
Lexeme
46904...
Lexeme
37020...
Lexeme
31979...
STRING STORE
46904...
"I"
37020...
"love"
31979...
"coffee"

VOCAB2INDEX

```
# counts per word
# some more cleaning should be done here
counts = Counter()
for cap_id in annIds:
    caption = coco.anns[cap_id]['caption']
    counts.update(spacy_tok(caption.lower()))
```

```
len(counts.keys())
```

```
23476
```

```
# delete words with little counts
for word in list(counts):
    if counts[word] < 5:
        del counts[word]
```

```
len(counts.keys())
```

```
8787
```

```
vocab2index = {"":0, "UNK":1, "<start>":2, "<end>": 3}
words = ["", "UNK", "<start>", "<end>"]
for word in counts:
    vocab2index[word] = len(words)
    words.append(word)
```

```
with open(PATH/'vocab2index.pickle', 'wb') as handle:
    pickle.dump(vocab2index, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

ENCODE SEQUENCES

```
def encode_sentence(caption, vocab2index):
    x = spacy_tok(str(caption).lower())
    x = np.array([vocab2index.get(w, vocab2index["UNK"]) for w in x])
    x = np.concatenate(([vocab2index["<start>"]], x, [ vocab2index["<end>"]]))
    return x
```

```
cap_id = annIds[0]
caption = coco.anns[cap_id]['caption']
print(caption)
encode_sentence(caption, vocab2index)
```

```
A very clean and well decorated empty bathroom
array([ 2,  4,  5,  6,  7,  8,  9, 10, 11,  3])
```

Accel.AI

Dataset

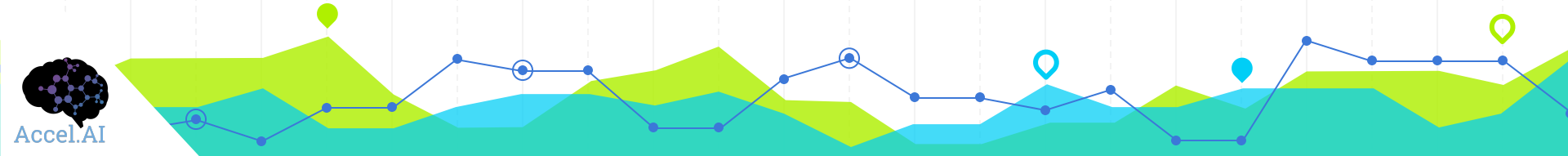
Normalize & Apply

4

Accel.AI

IMPORT DATA & DEFINE TRANSFORM FUNCTIONS

```
from torch.utils.data import Dataset
from torch.utils.data import DataLoader
import torch
import cv2

def apply_transforms(x, sz=250, zoom=1.05):
    """ Applies a random crop, rotation and flip"""
    sz2 = int(zoom*sz)
    x = cv2.resize(x, (sz2, sz2))
    x = random_crop(x, sz, sz)
    rdeg = (np.random.random()-.50)*20
    x = rotate_cv(x, rdeg)
    if np.random.random() > 0.5: x = np.fliplr(x).copy()
    return x

imagenet_stats = np.array([[0.485, 0.456, 0.406], [0.229, 0.224, 0.225]])

def normalize(im, stats=imagenet_stats):
    """Normalizes images with Imagenet stats."""
    return (im - stats[0])/stats[1]
```



COCO DATASET CLASS

```
class CocoDataset(Dataset):
    def __init__(self, path, json, vocab, transform=True, sz=250):
        self.path = path
        self.coco = COCO(json)
        self.ids = list(self.coco.anns.keys())
        self.vocab = vocab
        self.transform = transform
        self.sz = sz

    def __getitem__(self, index):
        ann_id = self.ids[index]
        caption = self.coco.anns[ann_id]['caption']
        img_id = self.coco.anns[ann_id]['image_id']
        filename = self.coco.loadImgs(img_id)[0]['file_name']
        x = read_image(self.path/filename)
        x = x/255
        # x = center_crop(x) # cropped in the resized
        # x = cv2.resize(x, (self.sz, self.sz)) # already resized
        if self.transform:
            x = apply_transforms(x)
        x = normalize(x)
        x = np.rollaxis(x, 2)
        return x, encode_sentence(caption, self.vocab)

    def __len__(self):
        return len(self.ids)
```

LOAD ANNOTATIONS & CREATE INDEX

```
with open(PATH/'vocab2index.pickle', 'rb') as handle:
    vocab = pickle.load(handle)
```

```
len(vocab.keys())
```

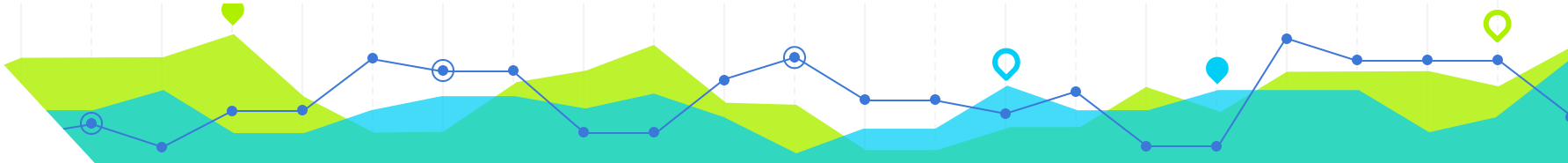
```
8791
```

```
! ls $PATH/"annotations"
```

```
captions_train2014.json  captions_val2014.json
```

```
train_ds = CocoDataset(PATH/"train_resized", PATH/"annotations/captions_train2014.json", vocab)
valid_ds = CocoDataset(PATH/"val_resized", PATH/"annotations/captions_val2014.json", vocab,
                       transform=False )
```

```
loading annotations into memory...
Done (t=0.79s)
creating index...
index created!
loading annotations into memory...
Done (t=0.29s)
creating index...
index created!
```



Accel.AI

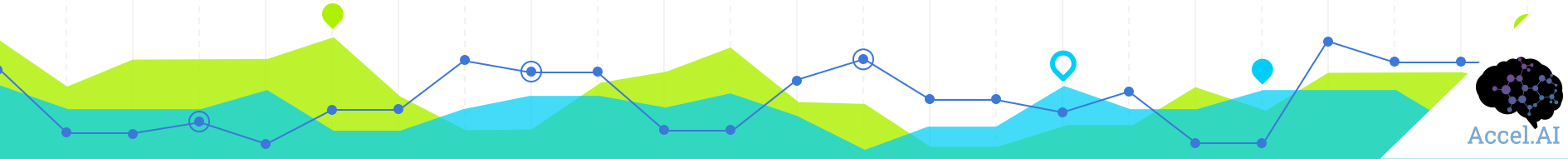
VALIDATE DATASET SHAPE & SIZE

```
x, y = valid_ds[20]
print(x.shape)
print(y)
```

```
(3, 250, 250)
[   2    4   11   60    7 1058  641 1694  745   19    3]
```

```
print(len(train_ds), len(valid_ds)) # 40k valid images
```

```
414113 202654
```



Accel.AI

VARIABLE LENGTH CAPTIONS

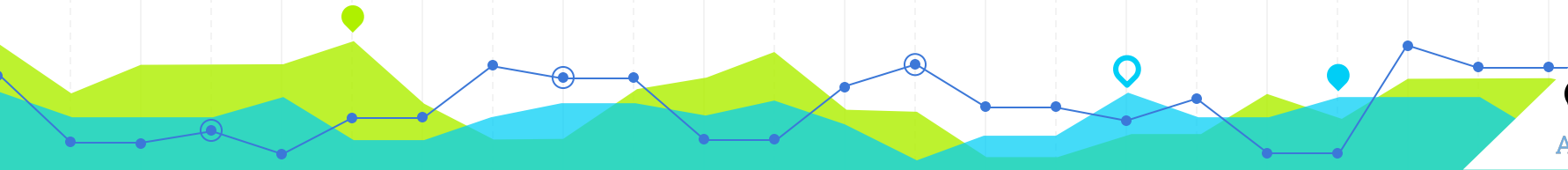
```python
def collate_fn(data):
    """Creates mini-batch tensors from the list of tuples (image, caption).

    Custom collate_fn for variable length padding.
    Args:
        data: list of tuple (image, caption).
            - image: numpy array of shape (3, 250, 250).
            - caption: list of word indexes of variable length.
    Returns:
        images: torch tensor of shape (batch_size, 3, 256, 256).
        targets: torch tensor of shape (batch_size, padded_length).
        lengths: list; valid length for each padded caption.
    """
    # Sort a data list by caption length (descending order).
    data.sort(key=lambda x: len(x[1]), reverse=True)
    print(data)
    images, captions = zip(*data)
    print("--------------------")
    print(images)

    # Merge images (from tuple of 3D tensor to 4D tensor).
    images = torch.stack(images, 0)

    # Merge captions (from tuple of 1D tensor to 2D tensor).
    lengths = [len(cap) for cap in captions]
    targets = torch.zeros(len(captions), max(lengths)).long()
    for i, cap in enumerate(captions):
        end = lengths[i]
        targets[i, :end] = torch.Tensor(cap[:end])
    return images, targets, lengths
```

LOAD TRAIN & VALIDATION SETS

```
trn_dl = DataLoader(train_ds, batch_size=2, shuffle=True, collate_fn=collate_fn)
val_dl = DataLoader(valid_ds, batch_size=2, shuffle=False, collate_fn=collate_fn)
```

```
x, y, lengths = next(iter(trn_dl))
```

```
[(array([[[-1.15259806, -1.06269719, -1.08185224, ...,  0.80716588,
           0.67787554,  0.67709528],
         [-1.15551891, -1.06205174, -1.08270602, ...,  0.41514463,
           0.65290738,  0.6682061 ],
         [-1.21845969, -1.03539017, -1.04873992, ...,  1.02948931,
           1.14246619,  1.10638247],
         ...,
         [-0.33650842, -0.33650842, -0.31954433, ..., -1.10776499,
          -1.37221525, -1.49619484],
         [-0.33201194, -0.33201194, -0.32113496, ..., -0.01517755,
          -0.29808874, -0.61940065],
         [-0.35964246, -0.35964246, -0.3263629 , ..., -0.01517755,
          -0.29808874, -0.61940065]],
```



VALIDATE SHAPE & SIZE

```
y
```

```
tensor([[    2,   372,   990,    55,   176,    83,     4,   116,   435,
           22,   113,    75,     4,   693,    19,     3],
        [    2,    95,    81,     4,    20,   858,   231,   441,     4,
          171,   802,    19,     3,     0,     0,     0],
        [    2,    54,  7371,  6047,    56,  7046,    81,     4,   443,
           14,   285,    19,     3,     0,     0,     0],
        [    2,    77,    55,  1841,     4,  2444,    38,     4,   590,
         1485,    19,     3,     0,     0,     0,     0],
        [    2,  1262,   134,   434,  7479,   105,    36,  1262,  3017,
           3,     0,     0,     0,     0,     0,     0]])
```

```
x.shape
```

```
torch.Size([5, 3, 250, 250])
```

```
lengths
```

```
[16, 13, 13, 12, 10]
```

Accel.AI

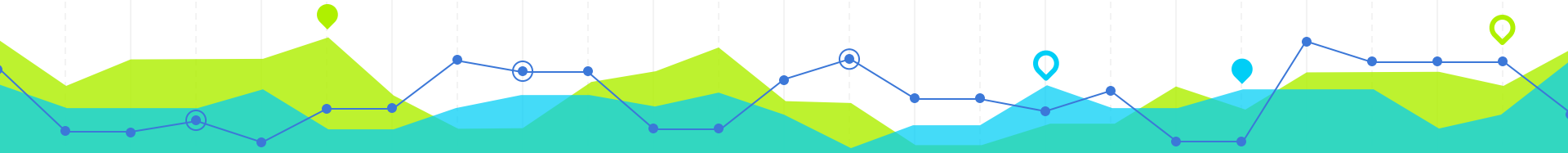


Image Captioning Models

Identifying & describing objects in images

5

Accel.AI

CHOOSING A MODEL

PyTorch Pretrained Models

https://pytorch.org/docs/stable/torchvision/models.html

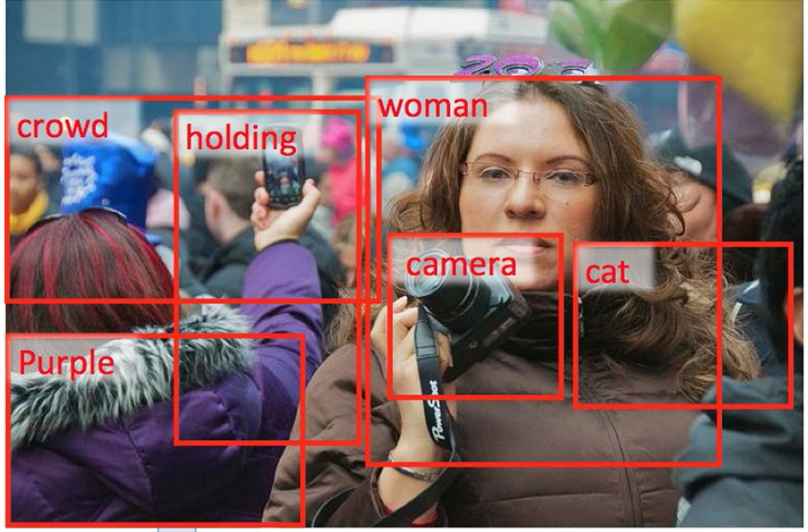
- AlexNet
- VGG
- ResNet
- SqueezeNet
- DenseNet
- Inception v3

Accel.AI

TRAINING YOUR OWN MODEL

- Millions of images in hundreds of categories
- Access to multiple GPUs
- A few weeks (2-3 for Image Net) to spare

Accel.AI



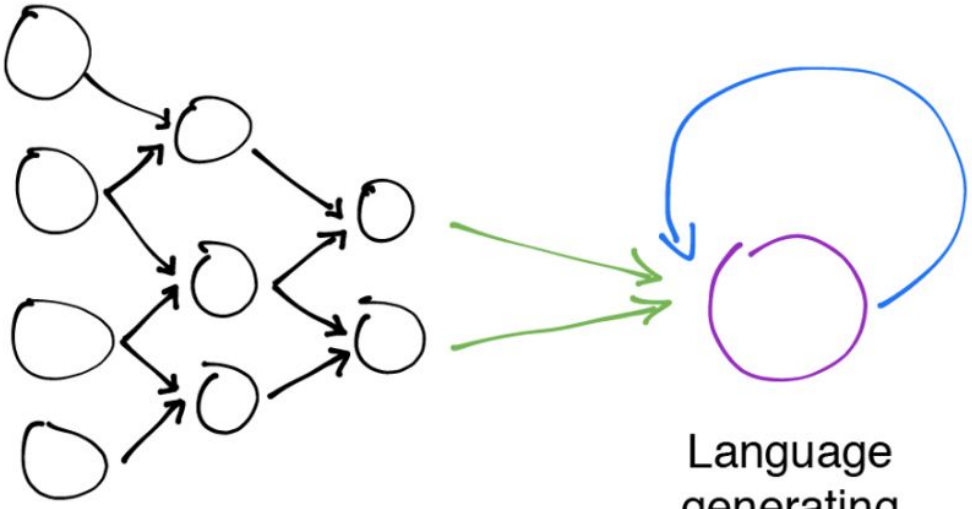
1. detect words → woman, crowd, cat, camera, holding, purple

2. generate sentences → A purple camera with a woman.
A woman holding a camera in a crowd.
...
A woman holding a cat.

3. re-rank sentences → #1 A woman holding a camera in a crowd.

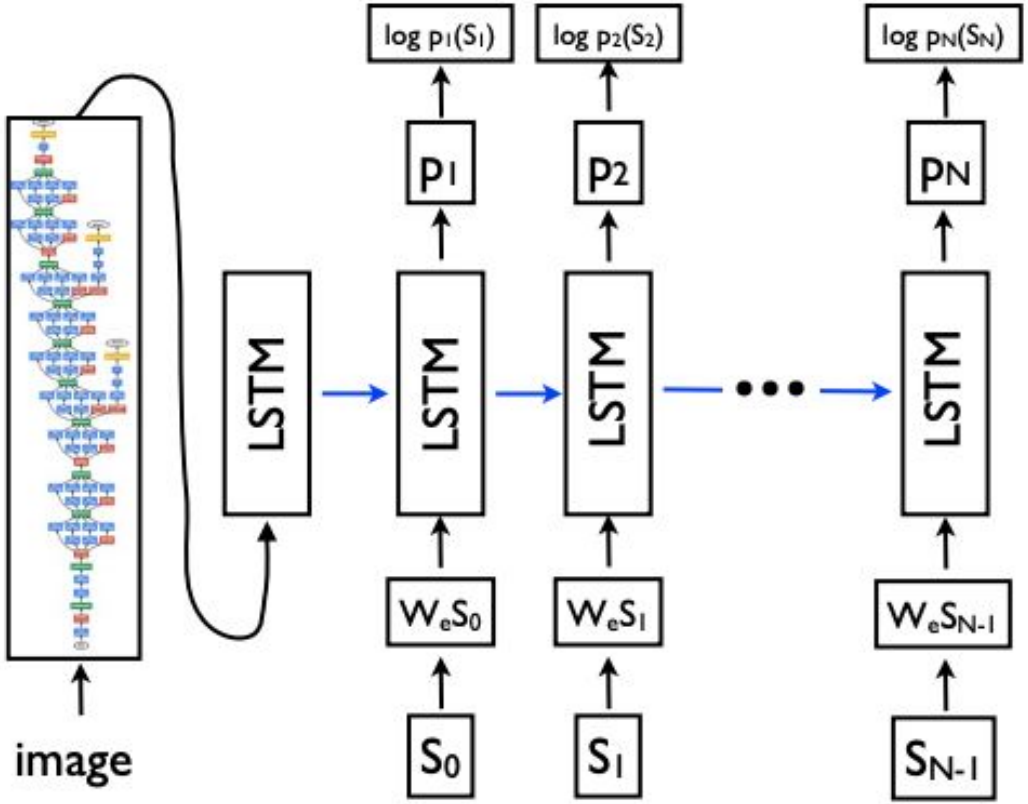
Microsoft Research blog

Vision

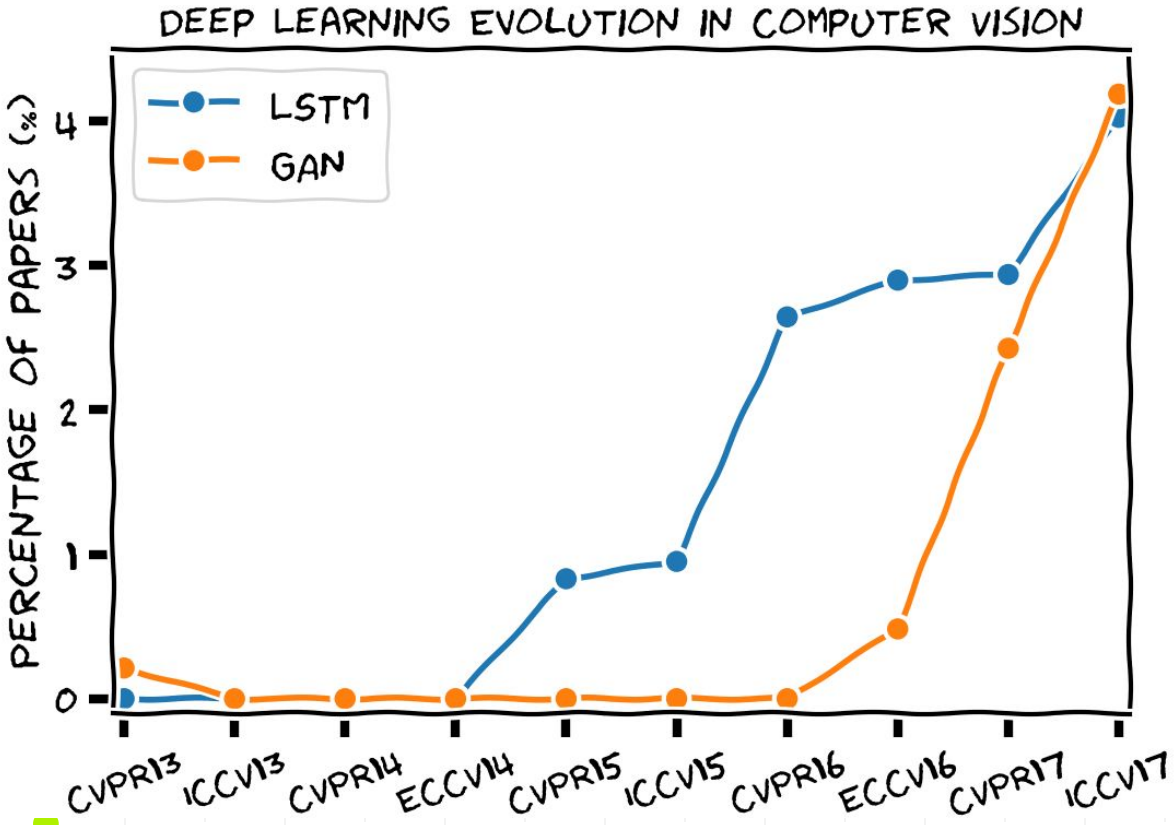


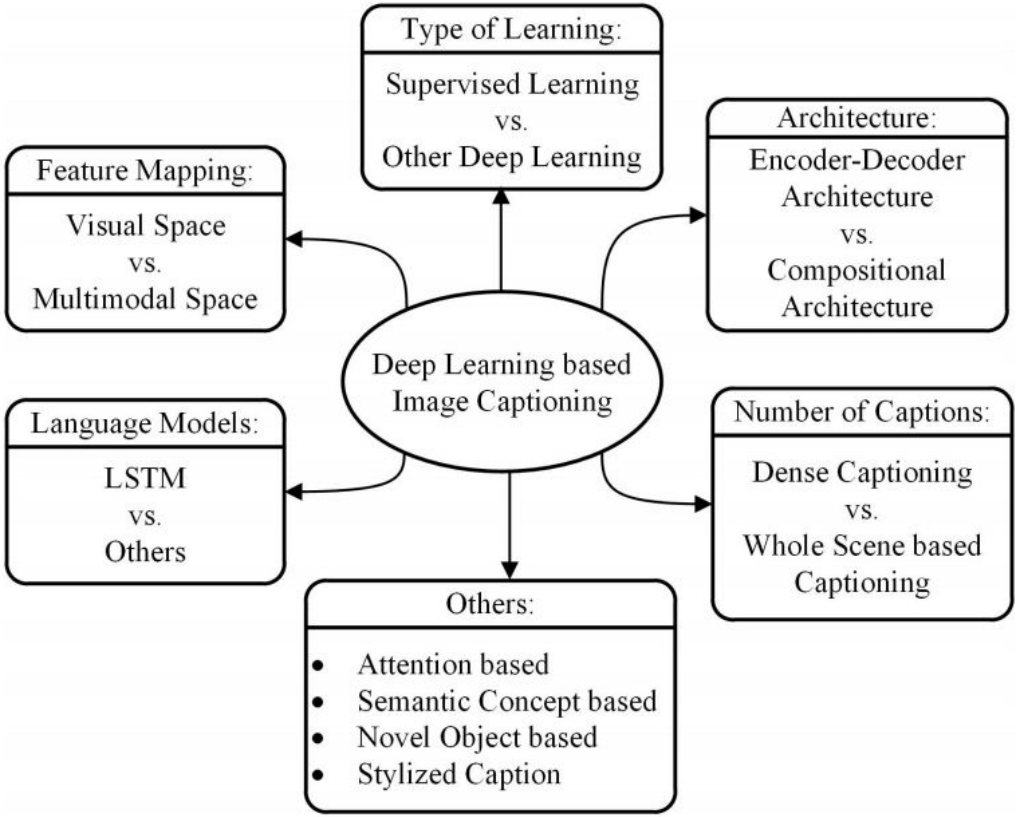
Deep CNN

Language generating RNN



Source: "Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge."


DEEP LEARNING EVOLUTION IN COMPUTER VISION
PERCENTAGE OF PAPERS (%)
0
1
2
3
4
LSTM
GAN
CVPR13
ICCV13
CVPR14
ECCV14
CVPR15
ICCV15
CVPR16
ECCV16
CVPR17
ICCV17

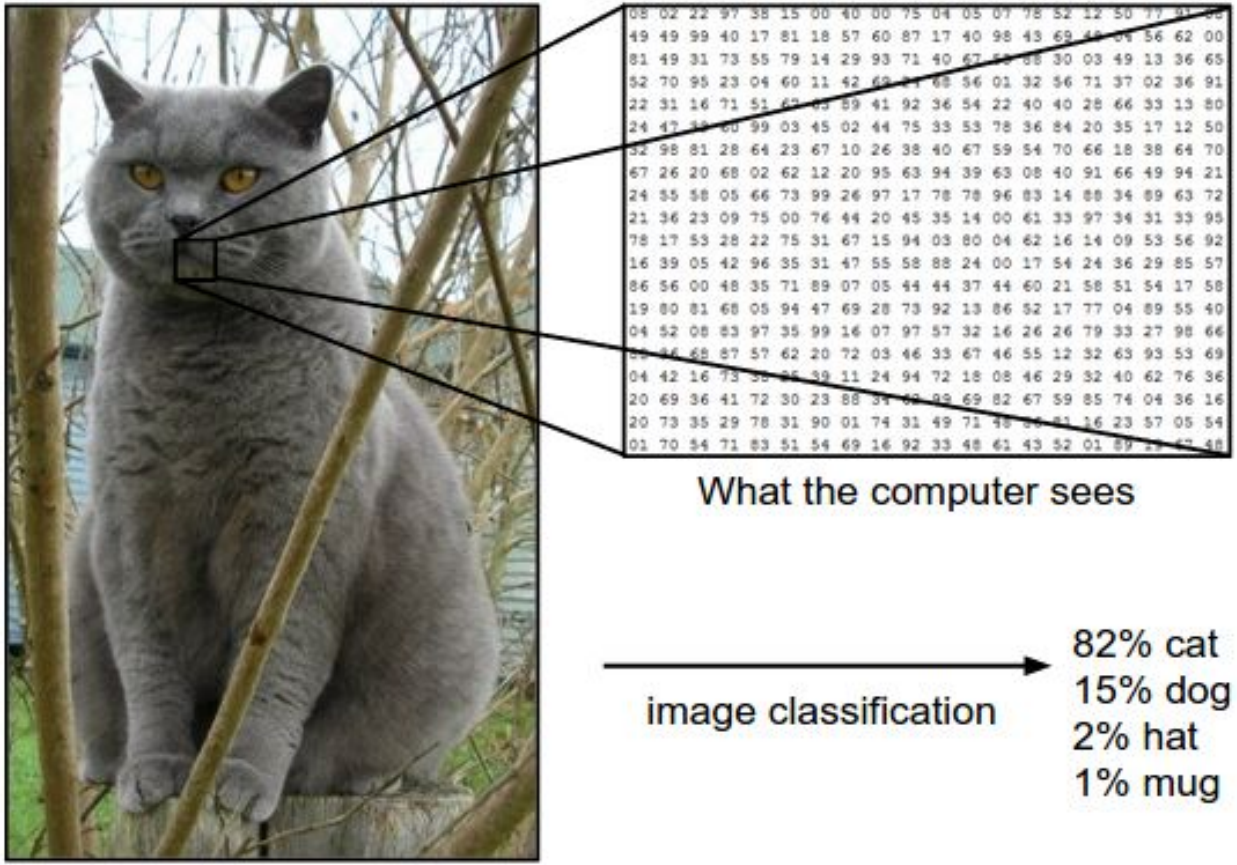

Deep Learning based Image Captioning
Type of Learning:
Supervised Learning
vs.
Other Deep Learning
Architecture:
Encoder-Decoder Architecture
vs.
Compositional Architecture
Feature Mapping:
Visual Space
vs.
Multimodal Space
Language Models:
LSTM
vs.
Others
Number of Captions:
Dense Captioning
vs.
Whole Scene based Captioning
Others:
Attention based
Semantic Concept based
Novel Object based
Stylized Caption

CNN FOR CV

ENCODER

6

Accel.AI

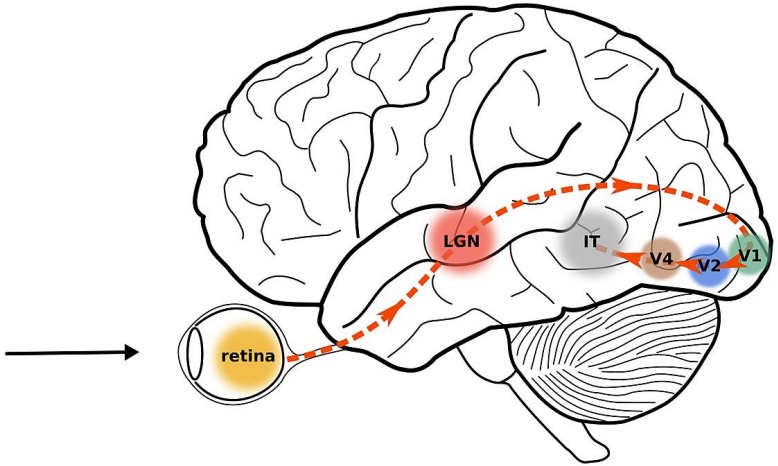


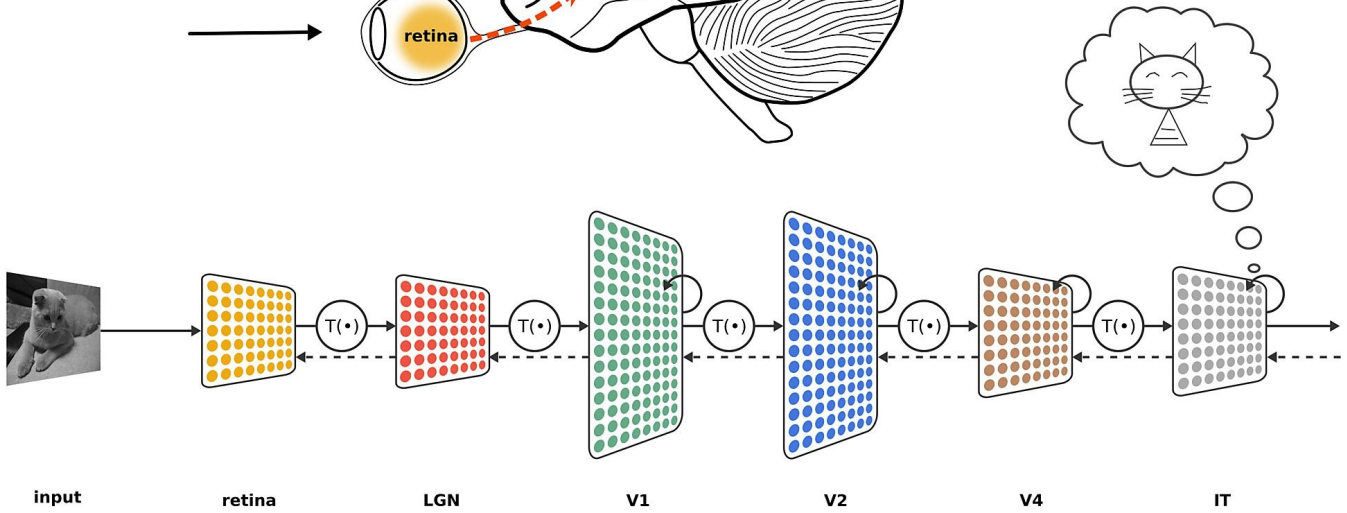
What the computer sees

image classification

82% cat
15% dog
2% hat
1% mug

Accel.AI





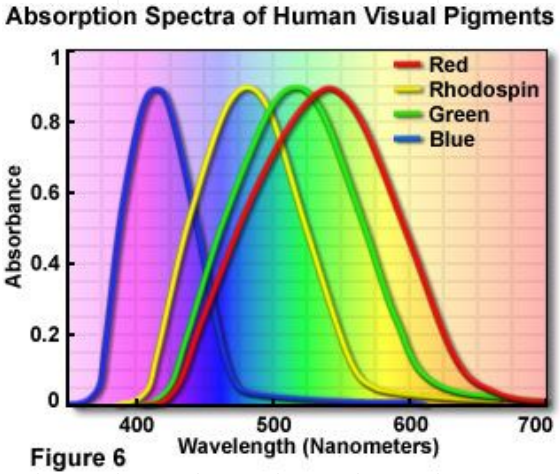
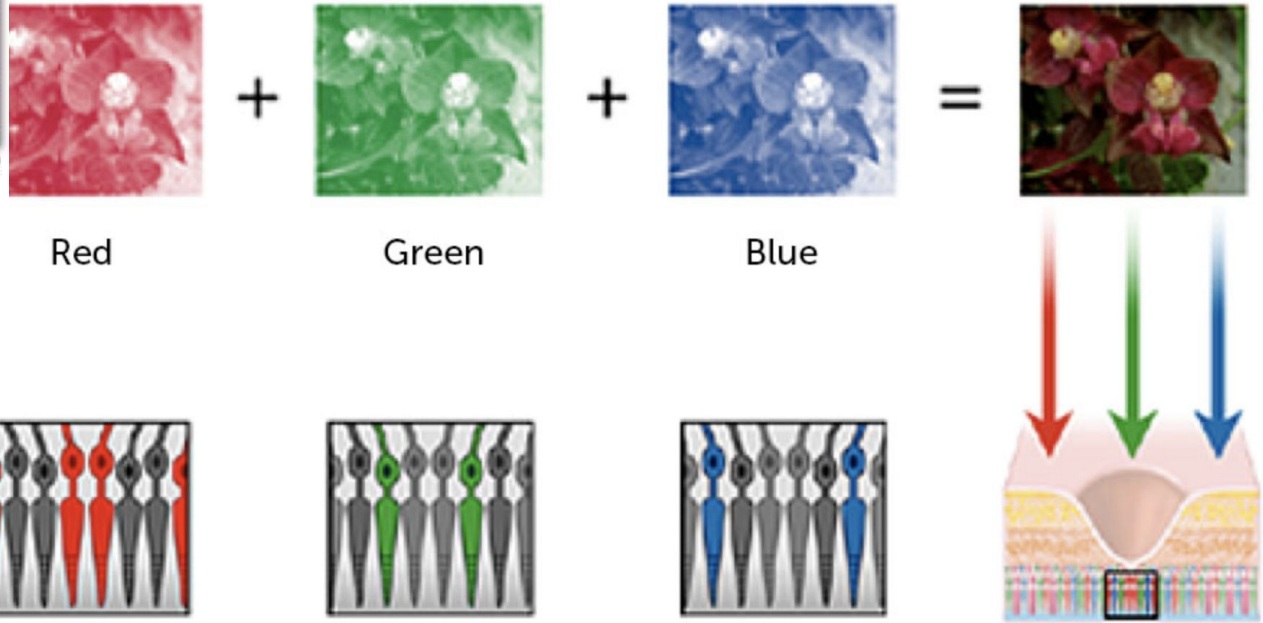
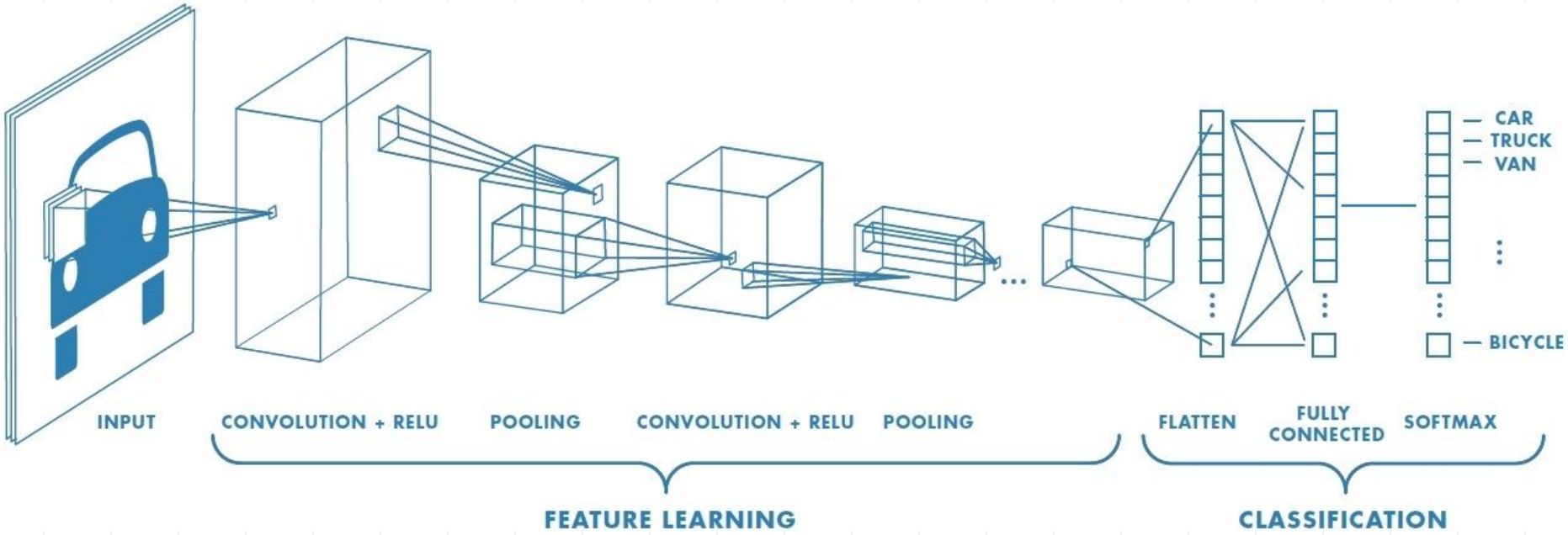


Figure 6



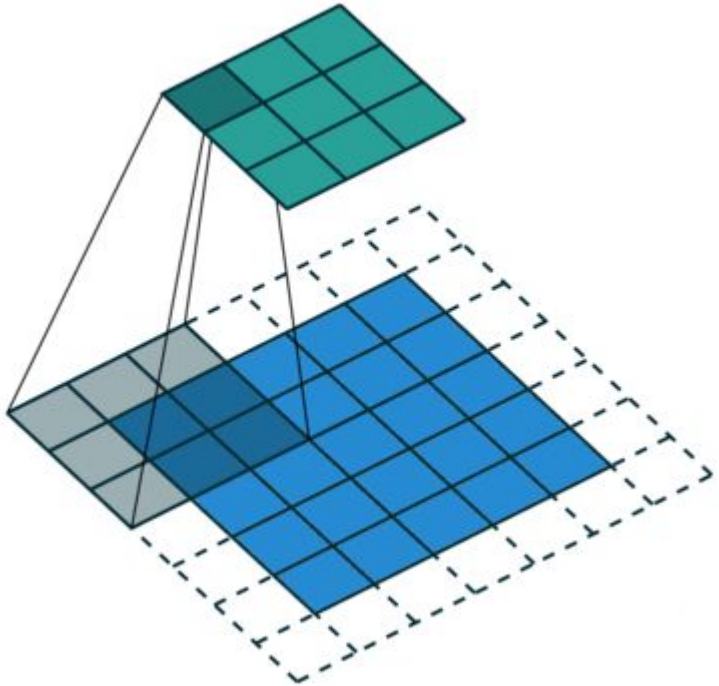
MIT neuroscientists find the brain can identify images seen for as little as 13 milliseconds

Accel.AI


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

Accel.AI

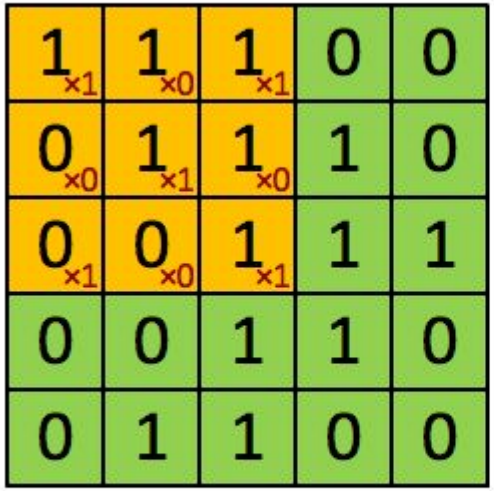
Convolutional Arithmetic

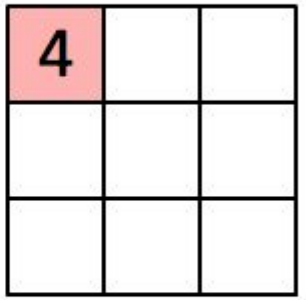


https://github.com/vdumoulin/conv_arithmetic

Accel.AI

Convolutional Arithmetic





Accel.AI

```
from torchvision.models import resnet18
```

```
class EncoderCNN(nn.Module):
    def __init__(self, embed_size):
        """Load the pretrained a resnet model and replace top fc layer."""
        super(EncoderCNN, self).__init__()
        resnet = models.resnet18(pretrained=True)
        modules = list(resnet.children())[:-2]      # delete the last 2 layers
        self.resnet = nn.Sequential(*modules)
        self.avgpool = nn.AdaptiveAvgPool2d(1)
        self.linear = nn.Linear(resnet.fc.in_features, embed_size)
        self.bn = nn.BatchNorm1d(embed_size, momentum=0.01)

    def forward(self, x):
        """Extract feature vectors from input images."""
        with torch.no_grad():
            x = self.resnet(x)
        x = self.avgpool(x)
        x = x.view(x.size(0), -1)
        x = self.bn(self.linear(x))
        return x
```

```
images, captions, lengths = next(iter(trn_dl))
```

```
embed_size = 100
encoder_model = EncoderCNN(embed_size).cuda()
```

```
encoder_model(images.cuda()).shape
```

```
torch.Size([5, 100])
```

VALIDATE CNN SIZE & SHAPE

```
images, captions, lengths = next(iter(trn_dl))
```

```
embed_size = 100
encoder_model = EncoderCNN(embed_size).cuda()
```

```
encoder_model(images.cuda()).shape
```

```
torch.Size([5, 100])
```

Accel.AI

RNN FOR NLP

DECODER

7

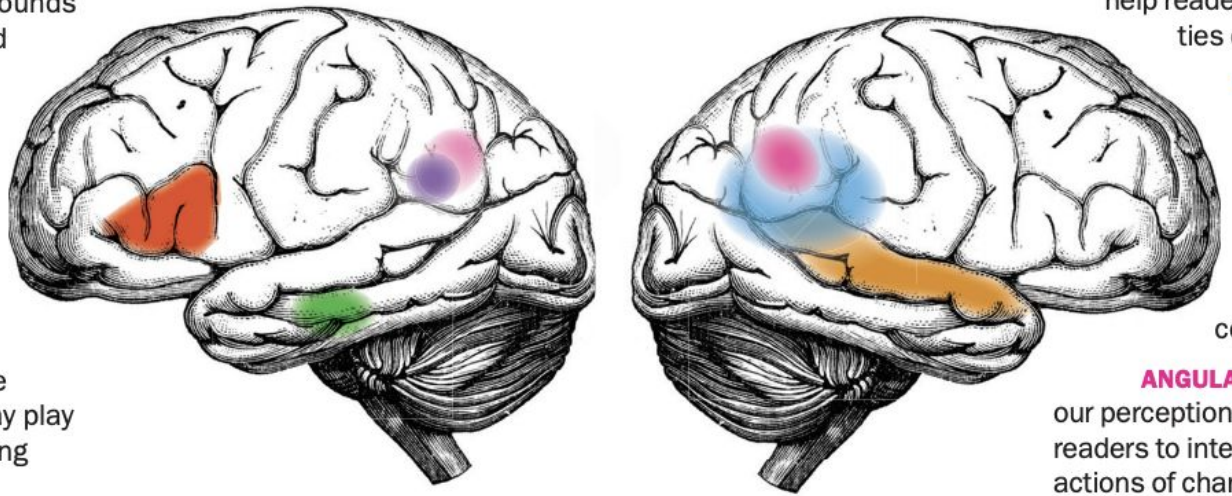
Accel.AI

Deciphering the Written Word

BROCA'S AREA: Known as the language center of the brain; creates the sounds of words in our head as we read.

VISUAL WORD FORM AREA: Allows us to recognize whole words as objects while reading.

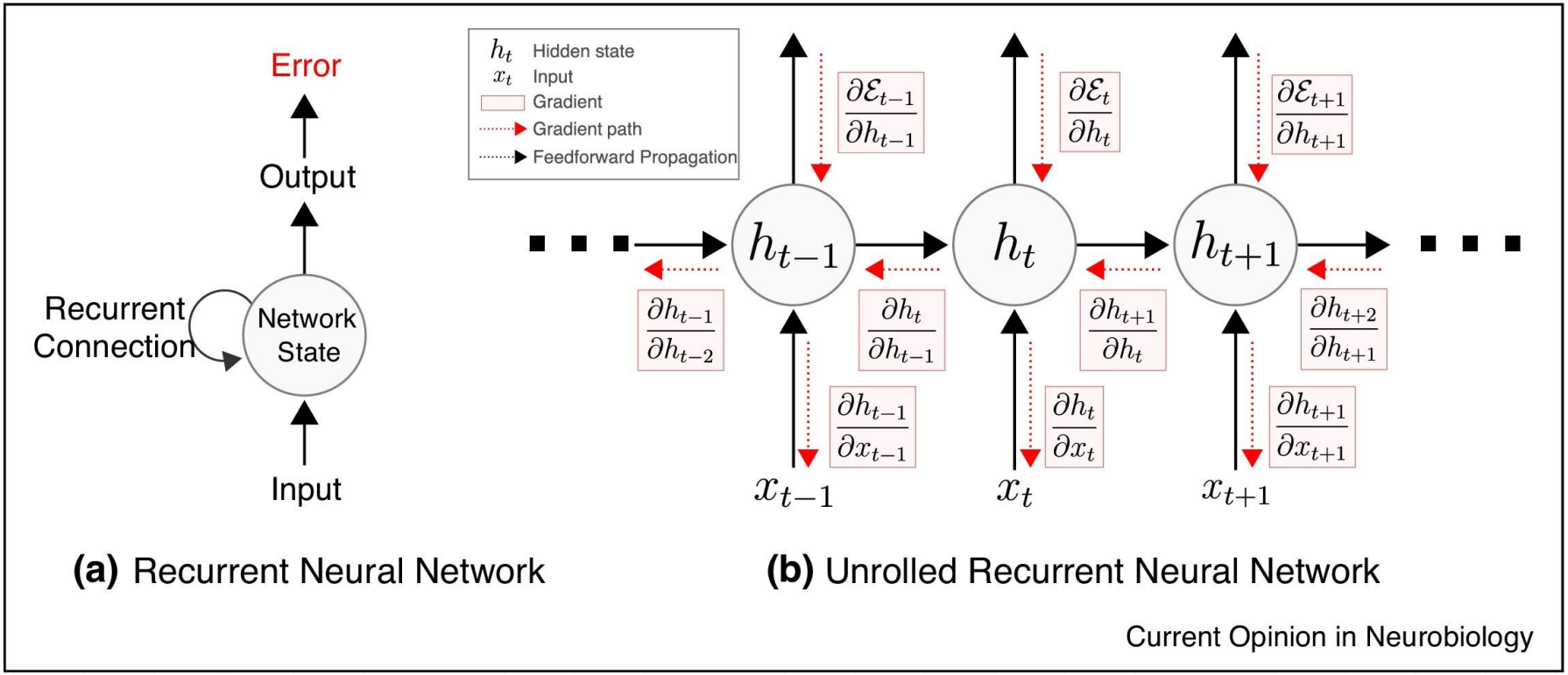
WERNICKE'S AREA: Involved in language comprehension; may play a role in remembering what we read.



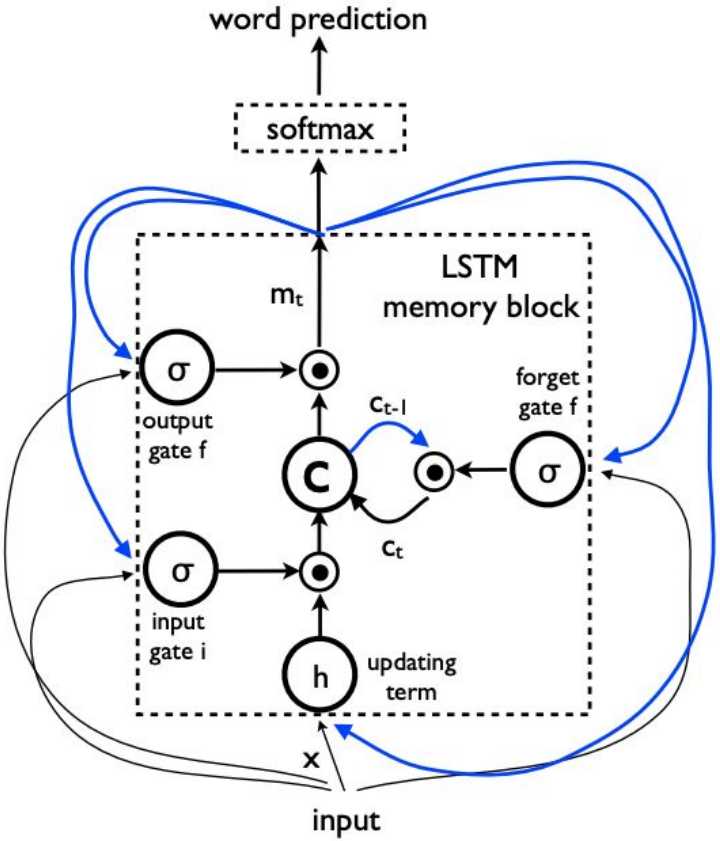
RIGHT POSTERIOR SUPERIOR TEMPORAL GYRUS: Facilitates theory of mind; may help readers distinguish the identities of story characters.

RIGHT PARIETO-TEMPORAL CORTEX: Found to be more active in good readers; may allow people to unpack longer and more complex sentences.

ANGULAR GYRUS: Involved in our perception of motion; may allow readers to interpret the physical actions of characters.

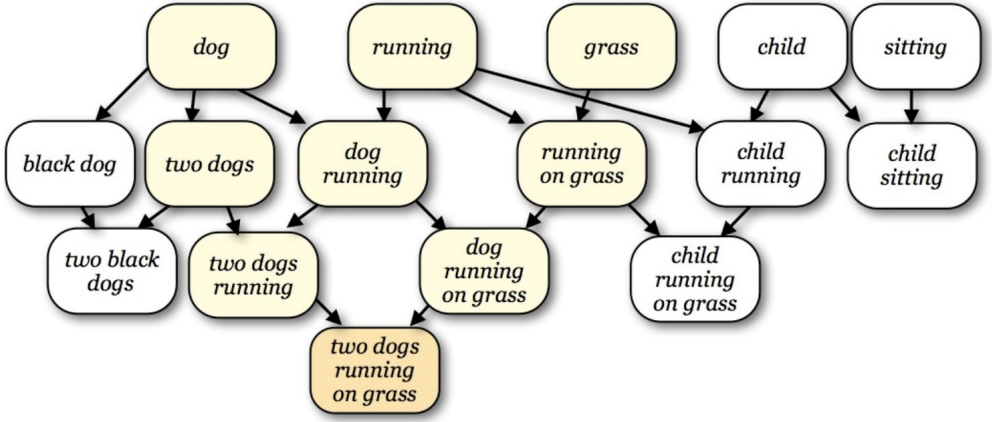


[Backpropagation through time and the brain, Fig 1]



$$
\begin{aligned}
i_t &= \sigma(W_{ix}x_t + W_{im}m_{t-1}) && (4)\\
f_t &= \sigma(W_{fx}x_t + W_{fm}m_{t-1}) && (5)\\
o_t &= \sigma(W_{ox}x_t + W_{om}m_{t-1}) && (6)\\
c_t &= f_t \odot c_{t-1} + i_t \odot h(W_{cx}x_t + W_{cm}m_{t-1}) && (7)\\
m_t &= o_t \odot c_t && (8)\\
p_{t+1} &= \mathrm{Softmax}(m_t) && (9)
\end{aligned}
$$

[Show and Tell: A Neural Image Caption Generator]



Each node in the graph corresponds to a string *s* and its denotation:

Flicker30K Denotation Graph

```
vocab_size = len(vocab.keys())
```

```
embed = nn.Embedding(vocab_size, embed_size)
```

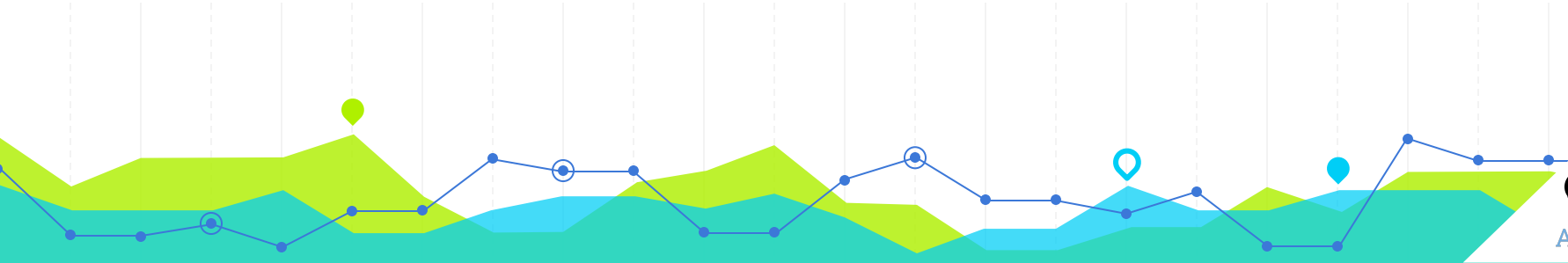
```
images, captions, lengths = next(iter(trn_dl))
```

```
embeddings = embed(captions.long())
embeddings.shape
```

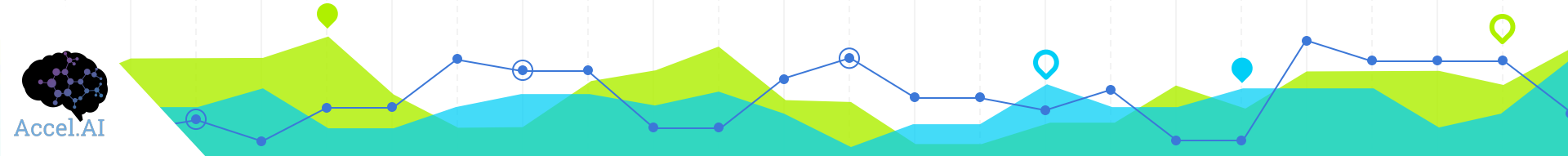
```
torch.Size([5, 21, 100])
```

```
images.shape
```

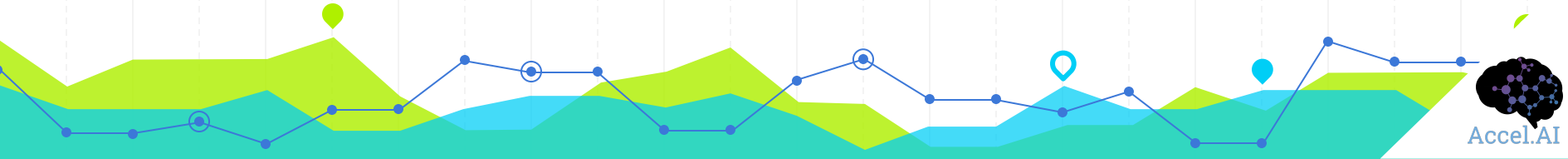
```
torch.Size([5, 3, 250, 250])
```



```python
class DecoderRNN(nn.Module):
    def __init__(self, embed_size, hidden_size, vocab_size, num_layers, max_seq_length=20):
        """Set the hyper-parameters and build the layers."""
        super(DecoderRNN, self).__init__()
        self.embed = nn.Embedding(vocab_size, embed_size)
        self.lstm = nn.LSTM(embed_size, hidden_size, num_layers, batch_first=True)
        self.linear = nn.Linear(hidden_size, vocab_size)
        self.max_seg_length = max_seq_length

    def forward(self, features, captions, lengths):
        """Decode image feature vectors and generates captions."""
        embeddings = self.embed(captions)
        embeddings = torch.cat((features.unsqueeze(1), embeddings), 1)
        packed = pack_padded_sequence(embeddings, lengths, batch_first=True)
        hiddens, _ = self.lstm(packed)
        outputs = self.linear(hiddens[0])
        return outputs
```



```python
    def sample(self, features, states=None):
        """Generate captions for given image features using greedy search."""
        sampled_ids = []
        inputs = features.unsqueeze(1)
        for i in range(self.max_seg_length):
            hiddens, states = self.lstm(inputs, states)          # hiddens: (batch_size, 1, hidden
_size)
            outputs = self.linear(hiddens.squeeze(1))            # outputs:  (batch_size, vocab_si
ze)
            _, predicted = outputs.max(1)                        # predicted: (batch_size)
            sampled_ids.append(predicted)
            inputs = self.embed(predicted)                       # inputs: (batch_size, embed_siz
e)
            inputs = inputs.unsqueeze(1)                         # inputs: (batch_size, 1, embed_s
ize)
        sampled_ids = torch.stack(sampled_ids, 1)                # sampled_ids: (batch_size, max_s
eq_length)
        return sampled_ids
```

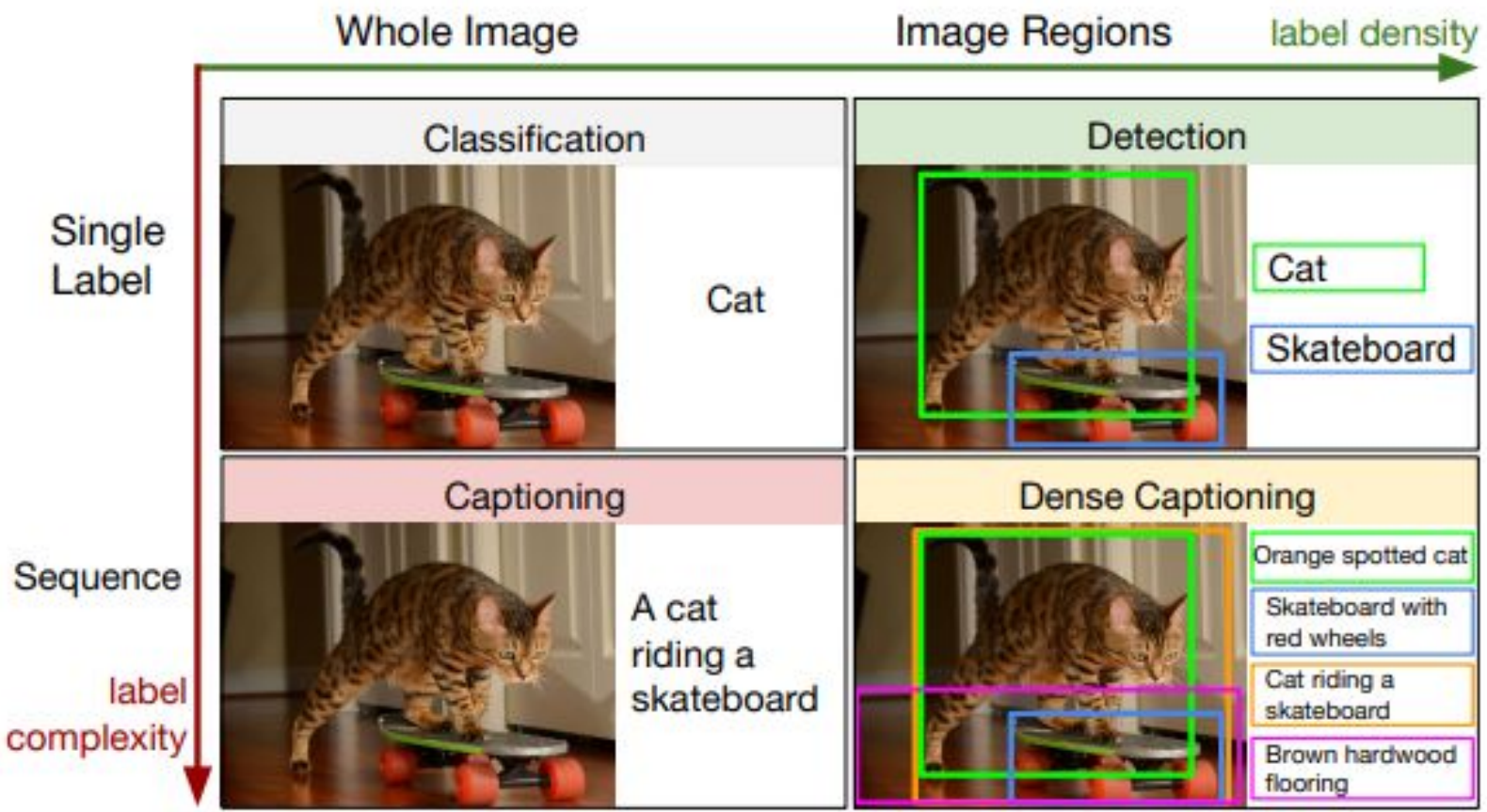


Training

CNN + RNN

8

Accel.AI


Whole Image
Image Regions
label density
Single Label
Sequence
label complexity
Classification
Cat
Detection
Cat
Skateboard
Captioning
A cat riding a skateboard
Dense Captioning
Orange spotted cat
Skateboard with red wheels
Cat riding a skateboard
Brown hardwood flooring

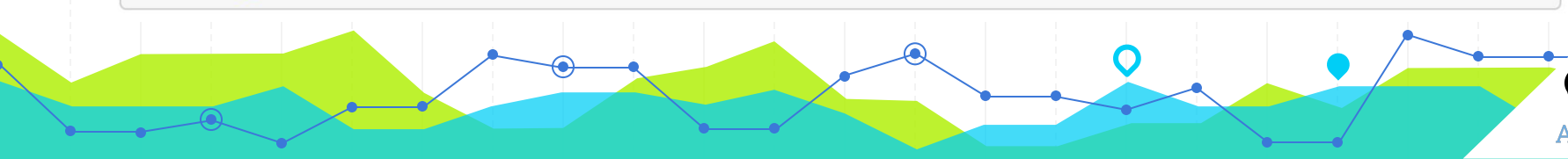
Training Setup

```
embed_size = 100
hidden_size = 512
num_layers = 1
batch_size = 128
```

```
encoder = EncoderCNN(embed_size).cuda()
decoder = DecoderRNN(embed_size, hidden_size, len(vocab), num_layers).cuda()

# Loss and optimizer
criterion = nn.CrossEntropyLoss()
params = list(decoder.parameters()) + list(encoder.linear.parameters()) + list(encoder.bn.paramete
rs())
optimizer = torch.optim.Adam(params, lr=0.001)
```

```
trn_dl = DataLoader(train_ds, batch_size=64, shuffle=True, collate_fn=collate_fn)
val_dl = DataLoader(valid_ds, batch_size=64, shuffle=False, collate_fn=collate_fn)
```

```
def save_model(m, p): torch.save(m.state_dict(), p)

def load_model(m, p): m.load_state_dict(torch.load(p))
```

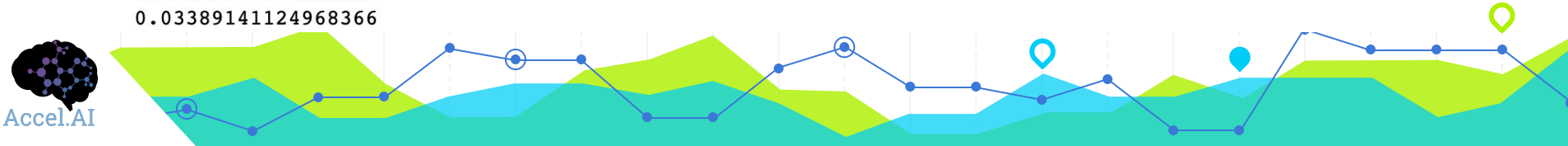


Accel.AI

```
def train(trn_dl, num_epochs=10):
    total_step = len(train_ds)
    for epoch in range(num_epochs):
        total_loss = 0
        for i, (images, captions, lengths) in enumerate(trn_dl):

            # Set mini-batch dataset
            images = images.cuda()
            captions = captions.cuda()
            targets = pack_padded_sequence(captions, lengths, batch_first=True)[0]

            # Forward, backward and optimize
            features = encoder(images)
            outputs = decoder(features, captions, lengths)
            loss = criterion(outputs, targets)
            decoder.zero_grad()
            encoder.zero_grad()
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
        save_model(encoder, PATH/"encoder-tmp.pth")
        save_model(decoder, PATH/"decoder-tmp.pth")
        print(total_loss/total_step)
```

```
train(trn_dl, num_epochs=10)
```

```
0.04012074929995369
0.03389141124968366
```



```
def save_model(m, p): torch.save(m.state_dict(), p)

def load_model(m, p): m.load_state_dict(torch.load(p))
```

```
save_model(encoder, PATH/"encoder-10.pth")
save_model(decoder, PATH/"decoder-10.pth")
```

```
train(trn_dl, num_epochs=3)
```

```
0.027335315742625047
0.02708266547260087
0.026858651942852918
```

```
save_model(encoder, PATH/"encoder-13.pth")
save_model(decoder, PATH/"decoder-13.pth")
```

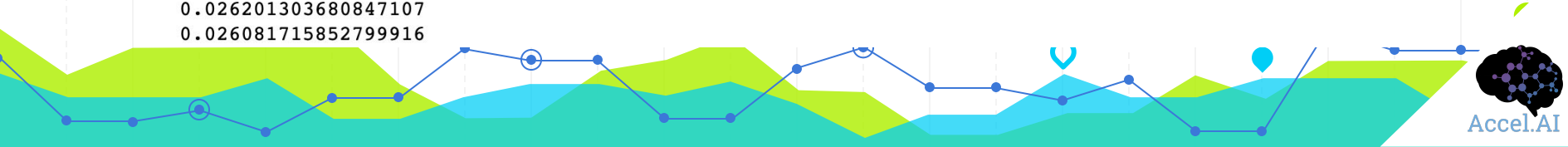
:
```
train(trn_dl, num_epochs=3)
```

```
0.026666308888500957
0.026493903817471923
0.02634520070580513
```

:
```
save_model(encoder, PATH/"encoder-16.pth")
save_model(decoder, PATH/"decoder-16.pth")
```

:
```
train(trn_dl, num_epochs=5)
```

```
0.026201303680847107
0.026081715852799916
```



Accel.AI

Evaluation

Validation & Loss

9

Accel.AI

```
def validation(val_dl, encoder, decoder):
    total_step = len(valid_ds)
    total_loss = 0
    for i, (images, captions, lengths) in enumerate(val_dl):
        images = images.cuda()
        captions = captions.cuda()
        targets = pack_padded_sequence(captions, lengths, batch_first=True)[0]

        features = encoder(images)
        outputs = decoder(features, captions, lengths)
        loss = criterion(outputs, targets)
        total_loss += loss.item()
    print(total_loss/total_step)
```

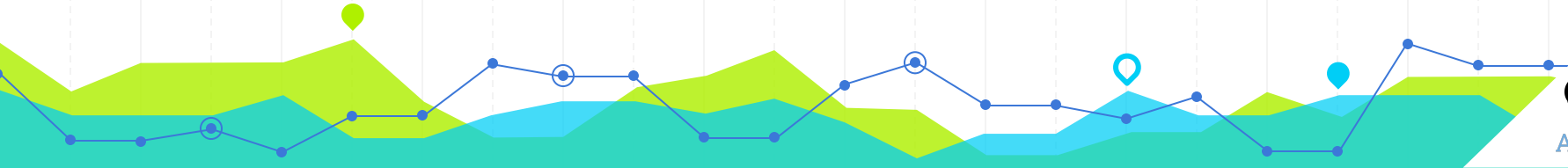
```
load_model(encoder, PATH/"encoder-40.pth")
```

```
load_model(decoder, PATH/"decoder-40.pth")
```

```
val_dl = DataLoader(valid_ds, batch_size=100, shuffle=False, collate_fn=collate_fn)
```

```
validation(val_dl, encoder, decoder)
```

```
0.25585410760106353
```



```
val_dl = DataLoader(valid_ds, batch_size=10, shuffle=True, collate_fn=collate_fn)
```

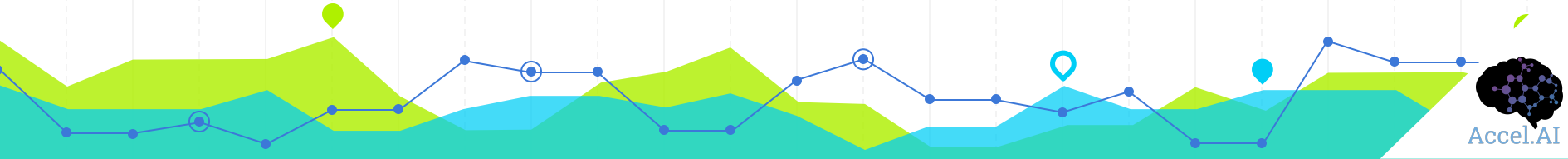
```
images, captions, lengths = next(iter(val_dl))
```

```
def clean_sentence(sentence_indices, words):
    sentence = []
    for i in sentence_indices:
        word = words[i]
        sentence.append(word)
        if word == "<end>":
            break
    return " ".join(sentence)
```

```
def denormalize(im, stats=imagenet_stats):
    """Normalizes images with Imagenet stats."""
    return im*stats[1] + stats[0]

def denorm(x):
    x = x.numpy()
    x = x.transpose(1,2,0)
    x = denormalize(x)
    return np.clip(x, 0 , 1)
```

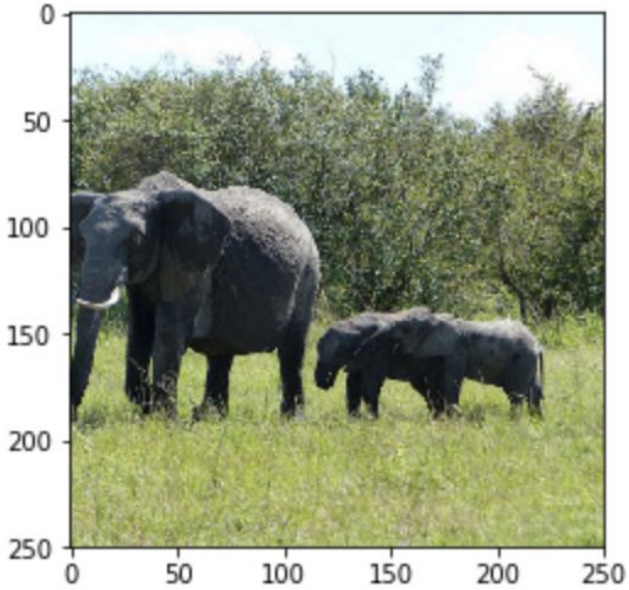
```
def look_at_results(val_dl, words):
    images, captions, lengths = next(iter(val_dl))
    features = encoder(images.cuda())
    pred_sentence = decoder.sample(features)
    for i in range(images.shape[0]):
        pred = clean_sentence(pred_sentence[i], words)
        actual = clean_sentence(captions[i].numpy(), words)
        print("Predicted:", pred)
        print("Actual:", actual, "\n")
        im = denorm(images[i])
        plt.imshow(im)
        plt.show()
```

```
look_at_results(val_dl, words)
```

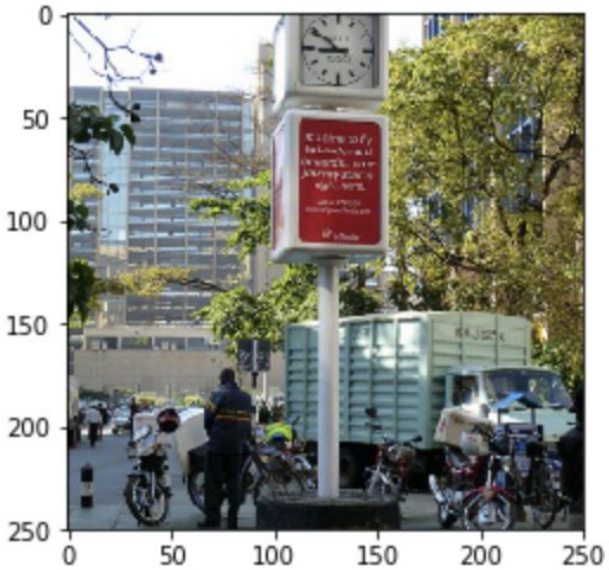


Predicted: <start> a man and a woman sitting at a table with a cake . <end>
Actual: <start> a boy getting ready to blow out candles on a birthday cake . <end>

Predicted: <start> a herd of elephants walking across a grass covered field . <end>
Actual: <start> an elephant and two baby elephants on the grassy field . <end>



Predicted: <start> a street sign on a bicycle rack on a street <end>
Actual: <start> a group of bicycles are parked under a clock <end>



Future Research & Applications

7

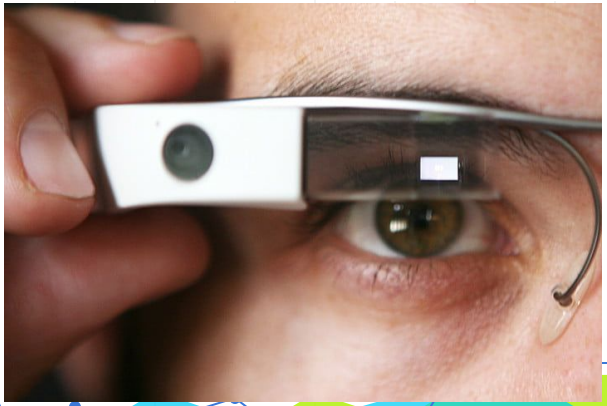
Image Captioning Tech

Accel.AI

Accurate Captions:

Now it looks like a cobbler that's been cooked.
These banana cakes won't be good until you put on your apron.

where characters such as Othello Hamlet and
Romeo and Juliet



Accel.AI

CURRENT RESEARCH

- Attention Models
  - Focus on select parts of the image
- GAN & RL
  - Diverse & Multiple Captions
- Dense Captioning
  - Region based image captioning

Accel.AI

EVALUATION METRICS

- BLEU
  - Small sentence evaluation
- ROGUE
  - Different types of text
- METEOR
  - Evaluates segments of captions
- SPICE
  - Understands semantic details

Accel.AI

Accel.AI

APPLIED AI LAB

Image Captioning in PyTorch
https://github.com/latinxinai/Intro-Image-Captioning-Lab/blob/master/image-caption-tutorial.ipynb

Lowering the barriers to entry in engineering artificial intelligence...

We focus on integrating AI and Social Impact through training, study sessions, and events with a focus on mindfulness and Applied AI engineering.

Our target audience includes underrepresented groups in tech, social justice advocates, and those experiencing job loss due to automation.

Accel.AI

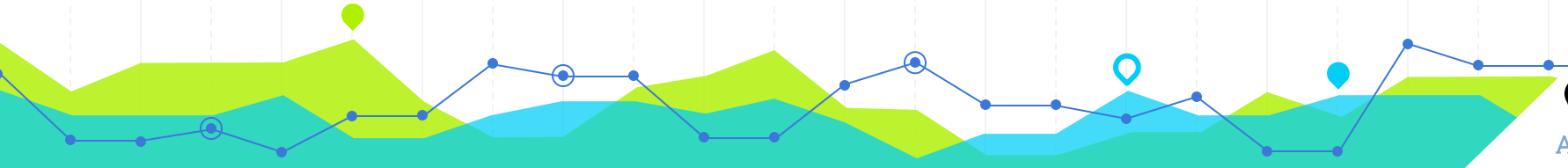
www.accel.ai | info@accel.ai

THANKS!

Any questions?

You can find me at

@quickresolute / info@accel.ai



Accel.AI

ADDITIONAL RESOURCES

Research

O. Vinyals, A. Toshev, S. Bengio and D. Erhan Show and Tell: A Neural Image Caption Generator, CVPR 2015

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollar, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John Platt, Larry Zitnick, Geoffrey Zweig, From Captions to Visual Concepts and Back, CVPR 2015

Md. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. 2018. A Comprehensive Survey of Deep Learning for Image Captioning. ACM Comput. Surv.

J. Wang and L. Perez, The Effectiveness of Data Augmentation in Image Classification using Deep Learning. 2017

Articles

Picture this: Microsoft Research project can interpret, caption photos

Videos

Andrej Karpathy - Automated Image Captioning with ConvNets and Recurrent Nets

Accel.AI

Join Us!

Accel AI Institute 501c3 Non Profit

info@accel.ai

@AccelerateAI

/accelai/

/accel.ai/

/company/accel.ai

www.accel.ai

www.latinxinai.org