Reminders

- HW1, due 2/1, 10pm
- Peer review for PRA1, due 2/2
- PRA2, due 2/8, 10pm

Artificial Intelligence Methods for Social Good

Lecture 6

Basics of Computer Vision

17-537 (9-unit) and 17-737 (12-unit)
Instructor: Fei Fang
feifang@cmu.edu

Learning Objectives

- Describe the concept of
  - Convolutional Neural Network
  - Cross-entropy loss
- For the poverty estimation problem, briefly describe
  - Significance/Motivation
  - Task being tackled, i.e., what is being solved/optimized
  - Model and method used to solve the problem
  - Evaluation process and criteria

Outline

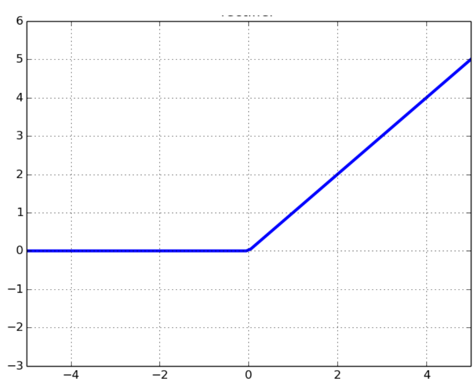
- Recap
- Convolutional Neural Network
- Estimate Poverty from Remote Sensing Data

Recap: A Basic Unit in Neural Networks

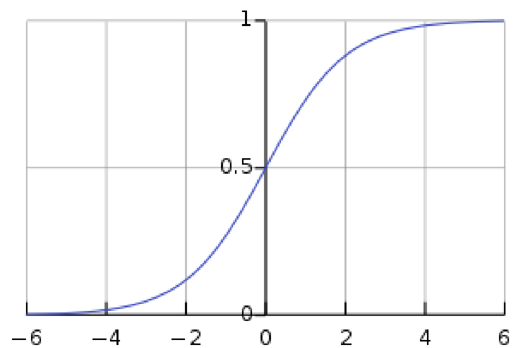
$$y = f(\boldsymbol{x}^T \boldsymbol{w} + b)$$

- Commonly used f
  - (default) ReLU (rectified linear unit): $f(z) = \max\{0, z\}$
  - Sigmoid: $f(z) = \sigma(z) = \frac{1}{1+e^{-z}}$
  - tanh: $f(z) = tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$

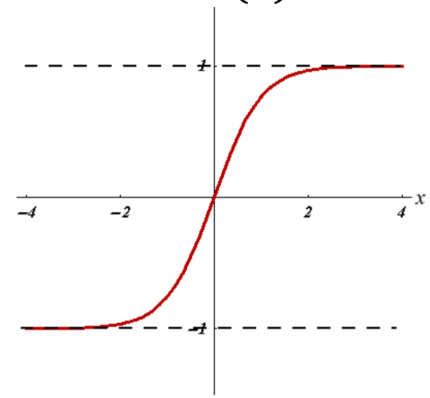
ReLU



$\sigma(z)$

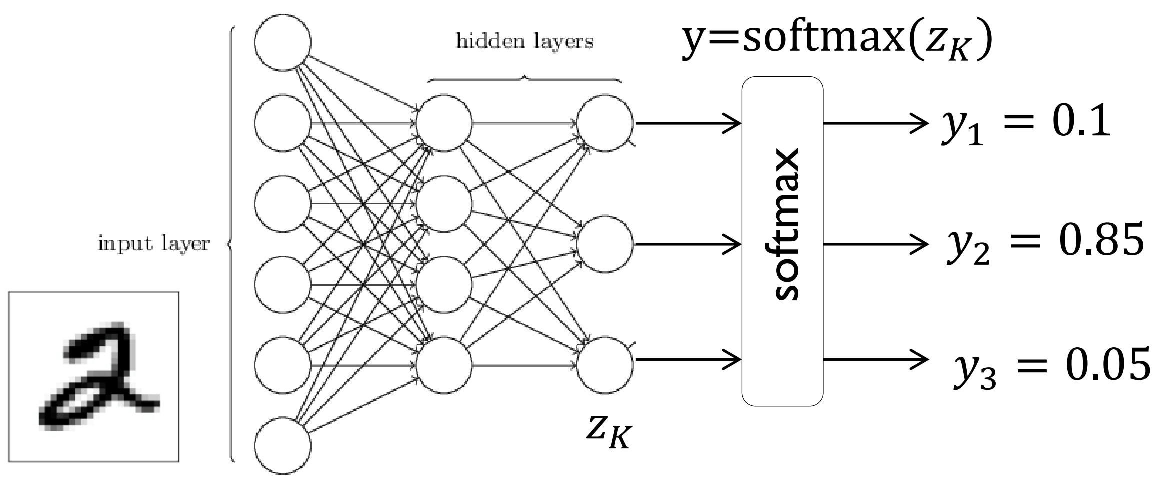


tanh(z)



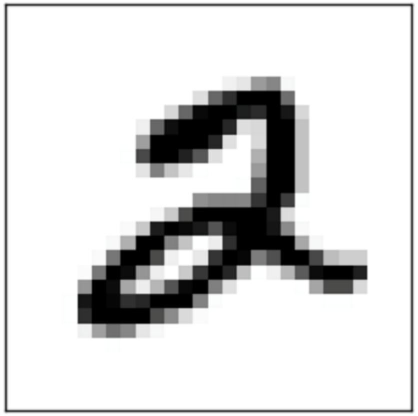
Recap: Feedforward Neural Network

- Fully connected layer / dense layer: every node in layer k is connected with every node in layer $k+1$
- MaxPool unit: $y = \max_{i \in \rightarrow y} x_i$
- Softmax Layer: $y_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$



Poll 1

- Given a set of grayscale images, each described by 28 by 28 pixels, if we treat the intensity of each pixel as an input feature, and the output is a single value representing the probability that the image is representing a handwritten digit 2, how many parameters are needed with one fully connected hidden layer with 100 nodes?
  - A: $28 \times 28 \times 100$
  - B: $(28 \times 28 + 1) \times 100$
  - C: $28 \times 28 \times 100 + 100$
  - D: $(28 \times 28 + 1) \times 100 + 101$
  - E: None of the above
  - F: I don't know

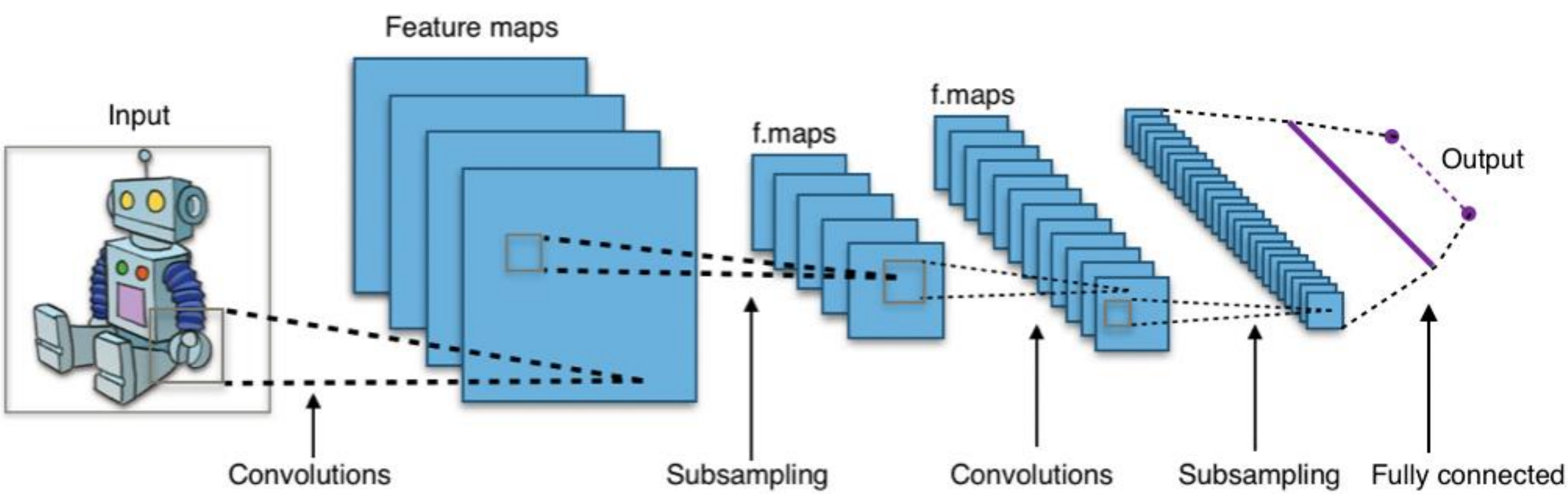


Outline

- Recap
- Convolutional Neural Network
- Estimate Poverty from Remote Sensing Data

Convolutional Neural Networks

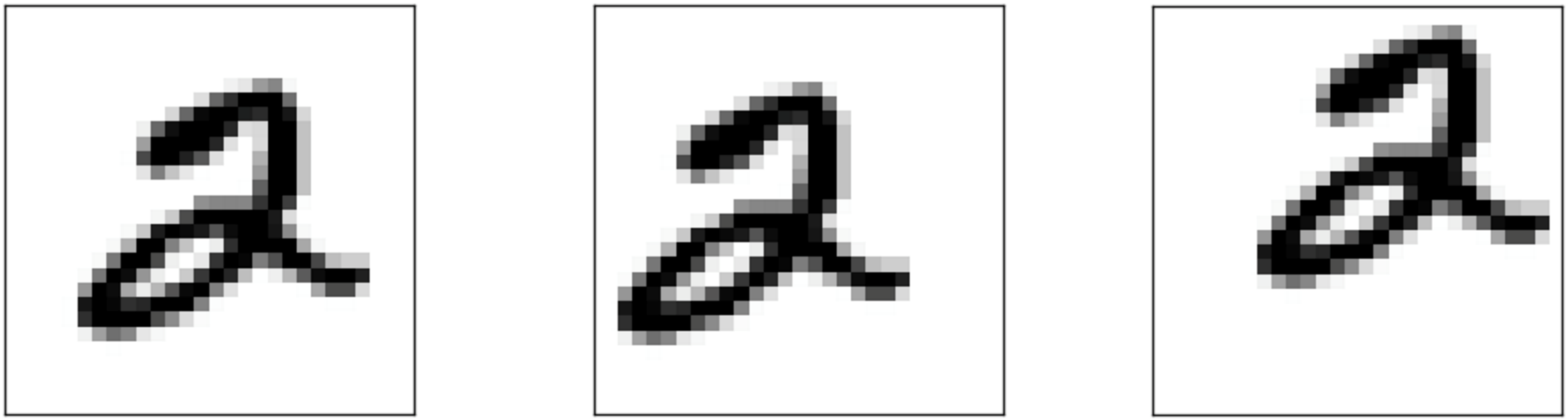
- A special type of FFN



Convolutional layer:
iteratively takes a matrix of inputs and outputs multiple (smaller) matrices
Pooling/Subsampling layer:
takes a matrix of inputs and subsamples those into a single element
e.g. by using MaxPool unit

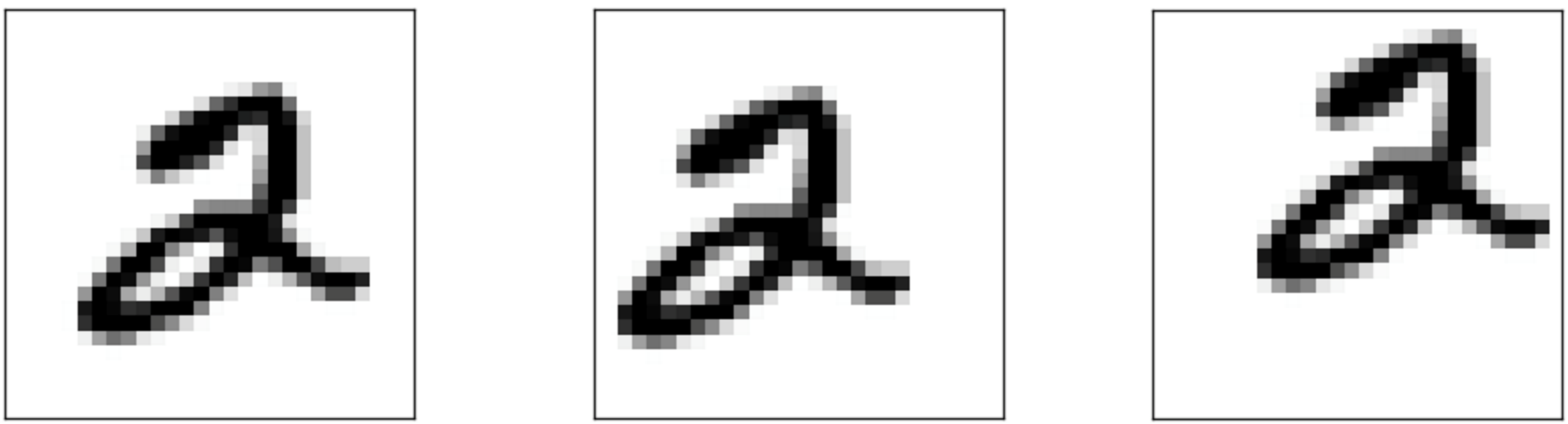
Convolutional Neural Networks

- How to represent the relationship between an input image and an output label (is it a hand written digit 2?) efficiently?



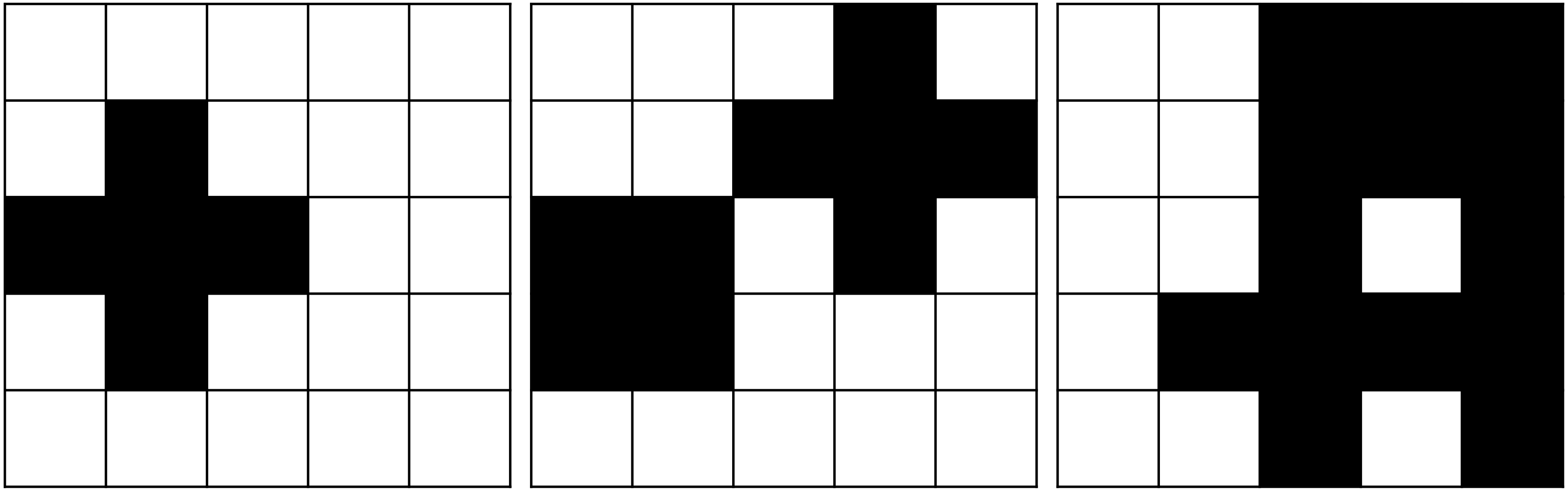
Convolutional Layer

- Motivation
  - Reduce parameters
  - Enforce shift invariance
- Key ideas
  - Construct a "filter", apply the filter to every subregion of the image (equivalently, sliding the filter)



Discussion

- Finding "+"
  - 5 by 5 pixels, B/W image, allow for noise in irrelevant area
- Construct an NN for the task using only basic units. How many hidden layers do you use?

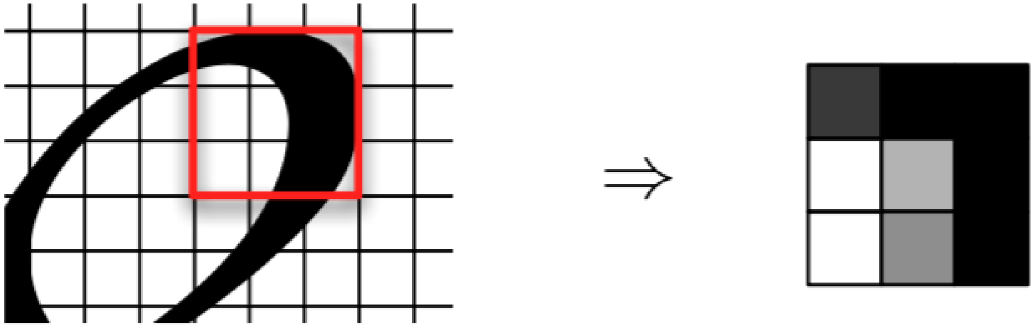


Convolutional Layer

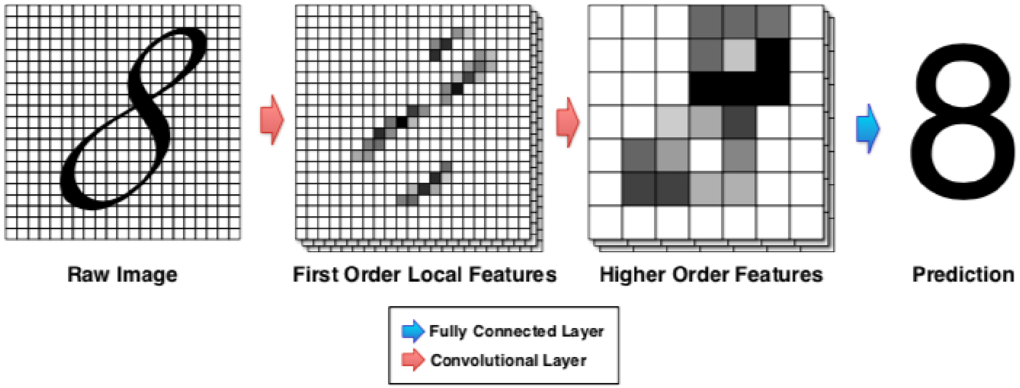
- Extend to general grayscale images
  - $z_{k+1} = z_k * w$ represent a set of nodes induced by one filter w
  - If w is a 3 by 3 matrix
    - $z_{k+1}^{11} = z_k^{11} w^{11} + z_k^{12} w^{12} + \cdots + z_k^{33} w^{33}$
    - $z_{k+1}^{12} = z_k^{12} w^{11} + z_k^{13} w^{12} + \cdots + z_k^{34} w^{33}$
    - $z_{k+1}^{ij} = z_k^{ij} w^{11} + z_k^{i,j+1} w^{12} + \cdots + z_k^{i+2,j+2} w^{33}$
    - …
  - Note: if we follow exact the standard convolution operation in image processing, it should be $z_{k+1}^{ij} = z_k^{ij} w^{33} + z_k^{i,j+1} w^{32} + \cdots + z_k^{i+2,j+2} w^{11}$, but it is equivalent to flipping the filter

Convolutional Layer

Convolutional window acts as a classifier for local features



Stacked convolutional layers learn higher level features



Convolutional Layer

- Extend to multi-band images (e.g., RGB images)
  - w is a 3 by 3 by #band matrix
- A single filter is not enough, often has many filters
  - Each filter leads to an output image

Convolutional Layer

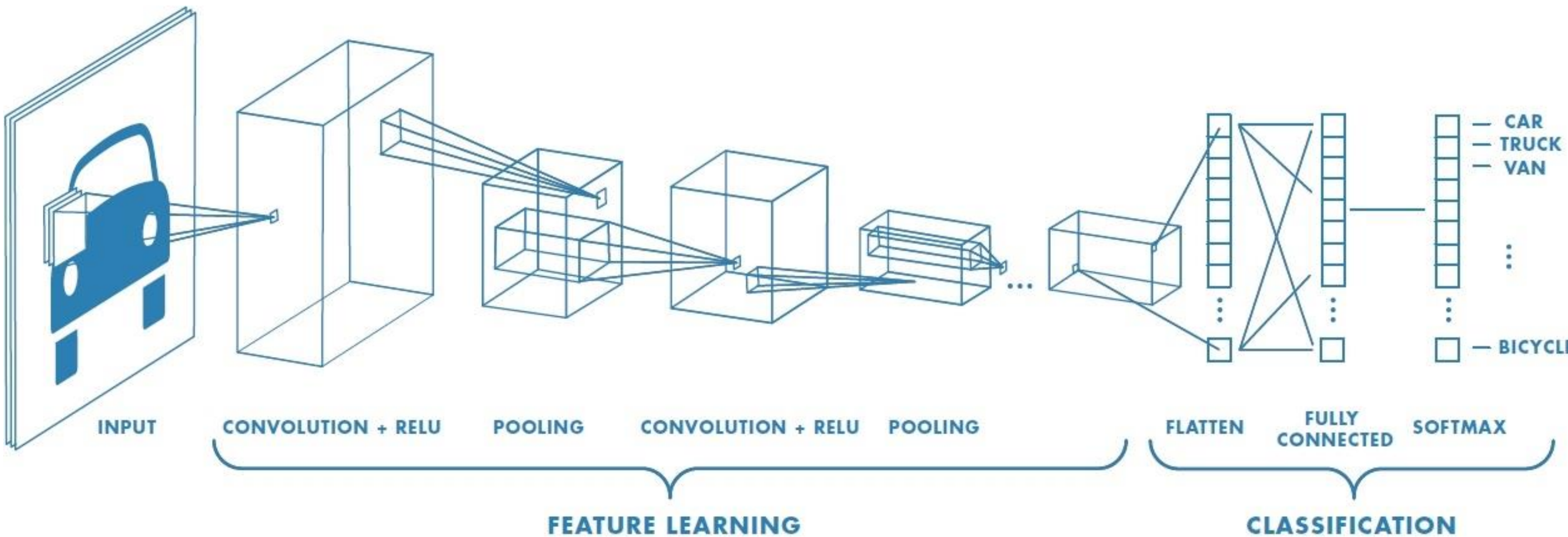
- Mathematical Convolution
  - $(f * g)(t) = \int_{\tau=-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$
- Convolution operation in images
  - Adding each element of the image to its local neighbors, weighted by the kernel

Convolutional Layer

- Sometimes zero-padding is used to get "images" of the same size in the next layer
- A convolutional layer is often followed by ReLU layer (element-wise) and pooling layer (region-wise) in image related tasks

Convolutional Neural Network

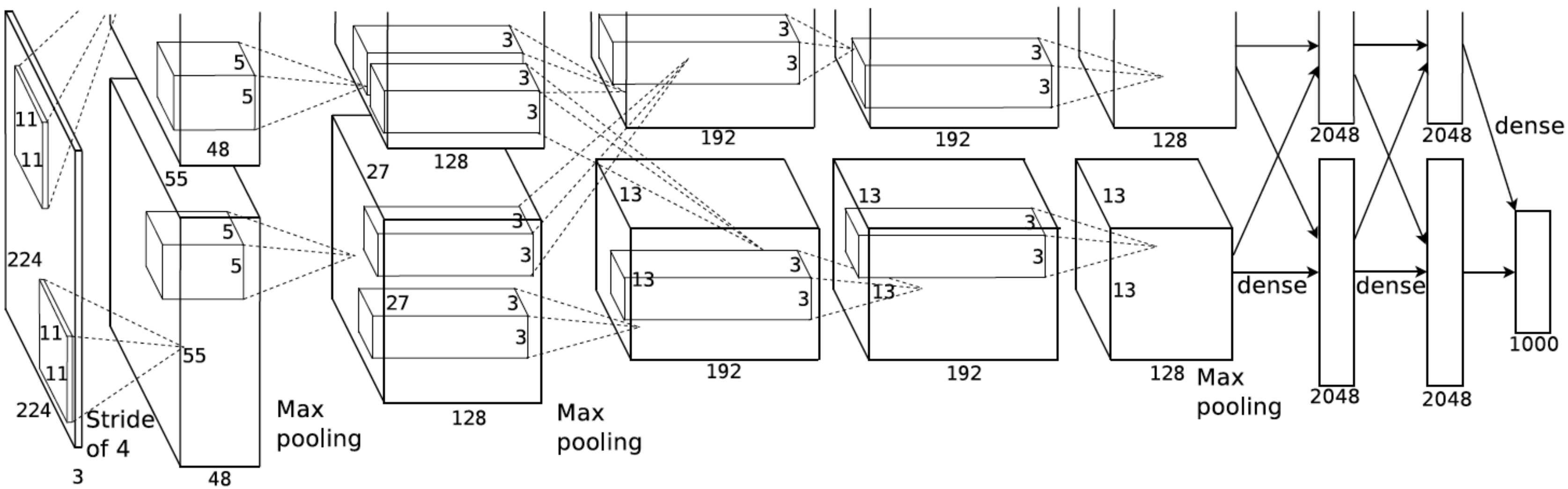
- CNN: An NN with convolutional layers



https://www.mathworks.com/discovery/convolutional-neural-network.html

Convolutional Neural Network

- AlexNet: Won image classification competition on ImageNet by a large margin (Krizhevsky, Sutskever, Hinton, 2012)



Cross Entropy Loss for Classification

- For classification task, output $f_\theta(x)$ is often a probability vector
  - Last layer is often a softmax layer
  - Outputs a vector that specifies the probability for each class
- Label y specifies a single class
- What loss function to use?
  - Cross entropy loss (or log loss): negation of log of output probability of the correct class

$$L(\theta) = \sum_{i=1}^{m} l(f_\theta(x^i), \hat{y}^i) = \sum_{i=1}^{m} -\log f_\theta(x^i)_{\hat{y}^i}$$

Train a CNN-based Image Classifier with PyTorch

- With Pytorch 1.2.0, MNIST dataset (grayscale, handwritten digits, 28*28 images)

```
class MyModel(nn.Module):
    def __init__(self):
        super(MyModel, self).__init__()

        # 28x28x1 => 26x26x32
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=32, kernel_size=3)
        self.d1 = nn.Linear(26 * 26 * 32, 128)
        self.d2 = nn.Linear(128, 10)

    def forward(self, x):
        # 32x1x28x28 => 32x32x26x26
        x = self.conv1(x)
        x = F.relu(x)

        # flatten => 32 x (32*26*26)
        x = x.flatten(start_dim = 1)

        # 32 x (32*26*26) => 32x128
        x = self.d1(x)
        x = F.relu(x)

        # logits => 32x10
        logits = self.d2(x)
        out = F.softmax(logits, dim=1)
        return out
```

One convolutional layer
32 output channels (32 filters)
3 by 3 square convolution kernel for each filter

Two fully connected layers

Use ReLU as activation function

Softmax layer for classification

https://colab.research.google.com/github/omarsar/pytorch_notebooks/blob/master/pytorch_quick_start.ipynb

Train a CNN-based Image Classifier with PyTorch

- Define hyperparameters in training and loss function

```
learning_rate = 0.001
num_epochs = 5

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model = MyModel()
model = model.to(device)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
```

Train a CNN-based Image Classifier with PyTorch

- Train the model

```
for epoch in range(num_epochs):
    train_running_loss = 0.0
    train_acc = 0.0

    model = model.train()

    ## training step
    for i, (images, labels) in enumerate(trainloader):

        images = images.to(device)
        labels = labels.to(device)

        ## forward + backprop + loss
        logits = model(images)
        loss = criterion(logits, labels)
        optimizer.zero_grad()
        loss.backward()

        ## update model params
        optimizer.step()

        train_running_loss += loss.detach().item()
        train_acc += get_accuracy(logits, labels, BATCH_SIZE)
```

Train a CNN-based Image Classifier with PyTorch

- Evaluate the trained model on test data

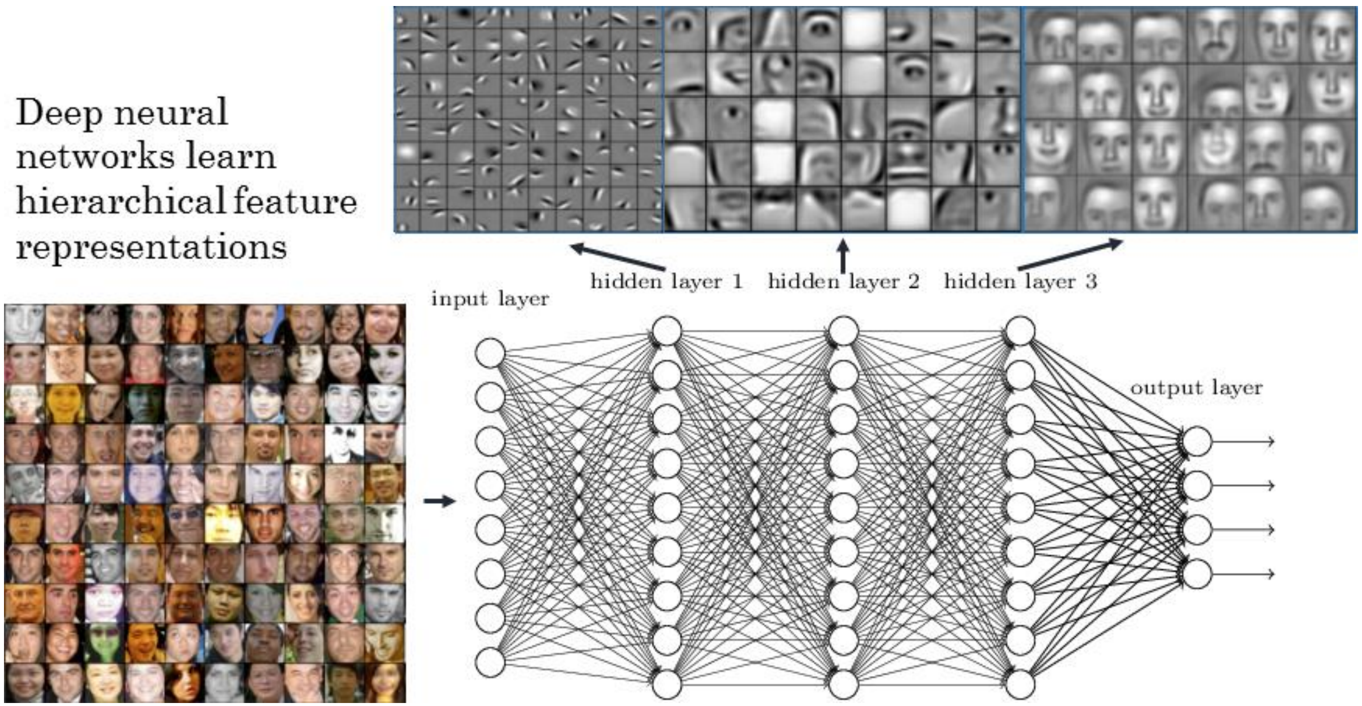
```
## compute accuracy
def get_accuracy(logit, target, batch_size):
    ''' Obtain accuracy for training round '''
    corrects = (torch.max(logit, 1)[1].view(target.size()).data == target.data).sum()
    accuracy = 100.0 * corrects/batch_size
    return accuracy.item()
```

```
test_acc = 0.0
for i, (images, labels) in enumerate(testloader, 0):
    images = images.to(device)
    labels = labels.to(device)
    outputs = model(images)
    test_acc += get_accuracy(outputs, labels, BATCH_SIZE)

print('Test Accuracy: %.2f'%( test_acc/i))
```

Reuse Lower Layers in CNN

- Lower layers are reusable!



https://www.rsipvision.com/exploring-deep-learning/

Reuse Lower Layers in CNN

- Typical workflow
  - Train a network NN_A for Problem A with dataset D_A
  - Build a network NN_B for Problem B, with same lower layer architecture
  - Initialize the lower layers parameters of NN_B using NN_A
  - Refine NN_B with dataset D_B for Problem B

Outline

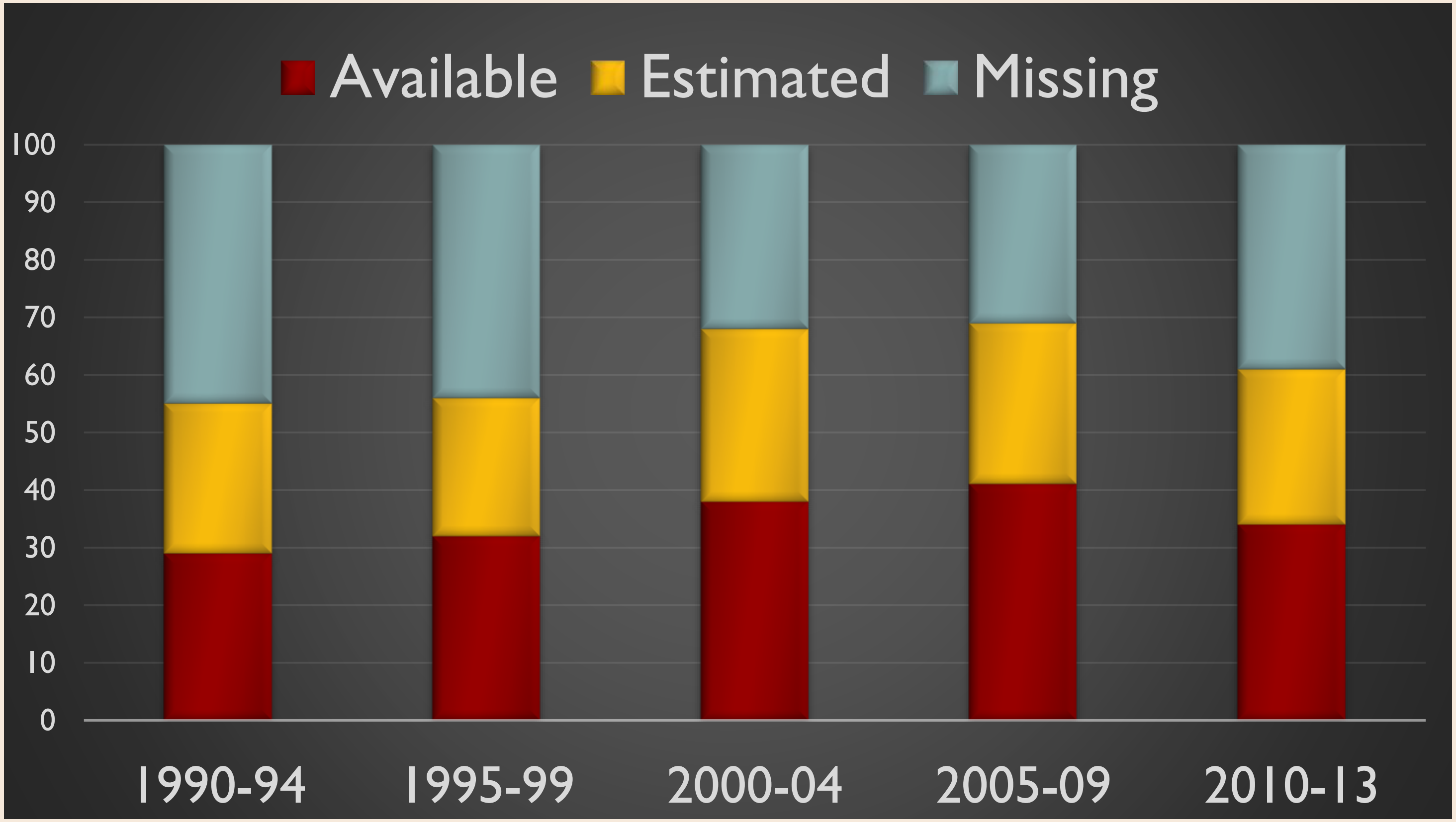
- Recap
- Convolutional Neural Network
- Estimate Poverty from Remote Sensing Data

Motivation


SUSTAINABLE DEVELOPMENT GOALS
1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS
SUSTAINABLE DEVELOPMENT GOALS

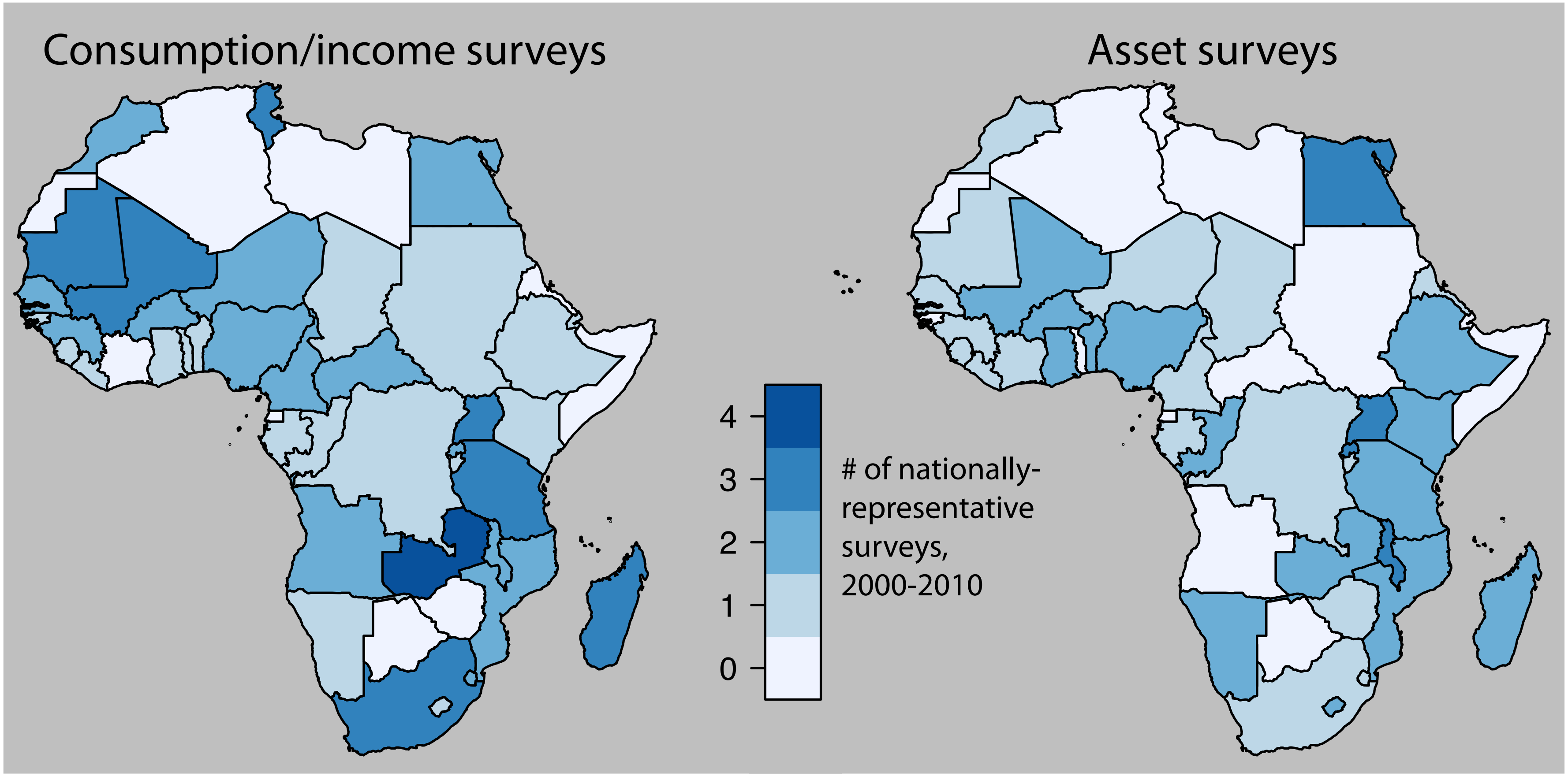
Data Gaps (Lacking Key Measures of Dev.)

Data for tracking development goals

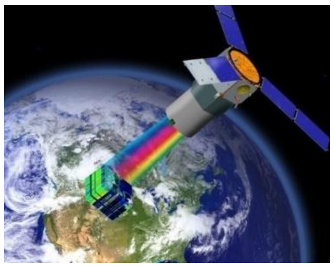


Source: United Nations Statistics Division

Previous Approach -- Survey



Data Revolution with AI



Technology Push → **Artificial Intelligence** → **Society Pull**

Data

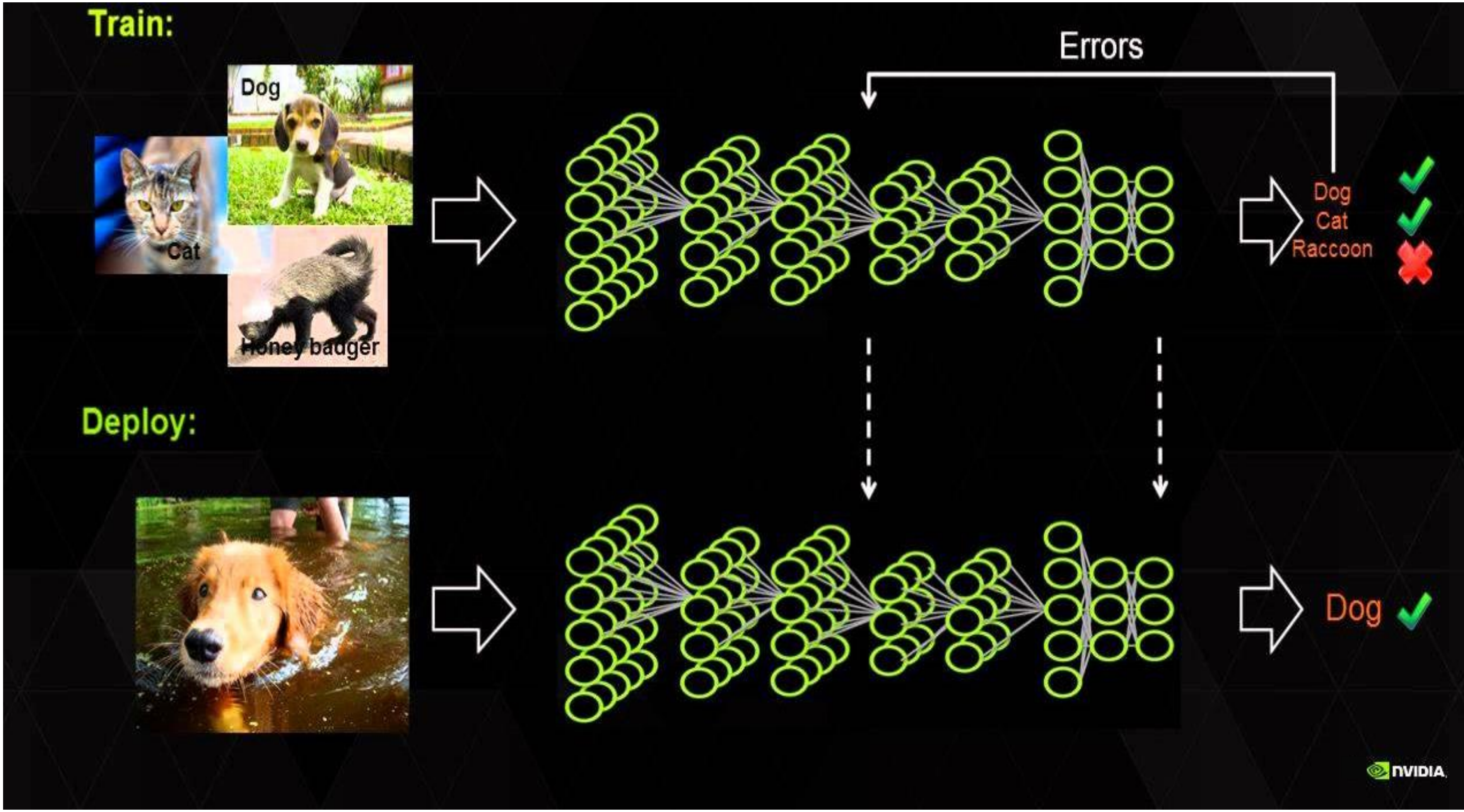
Insights

Remote sensing is becoming cheaper and more accurate

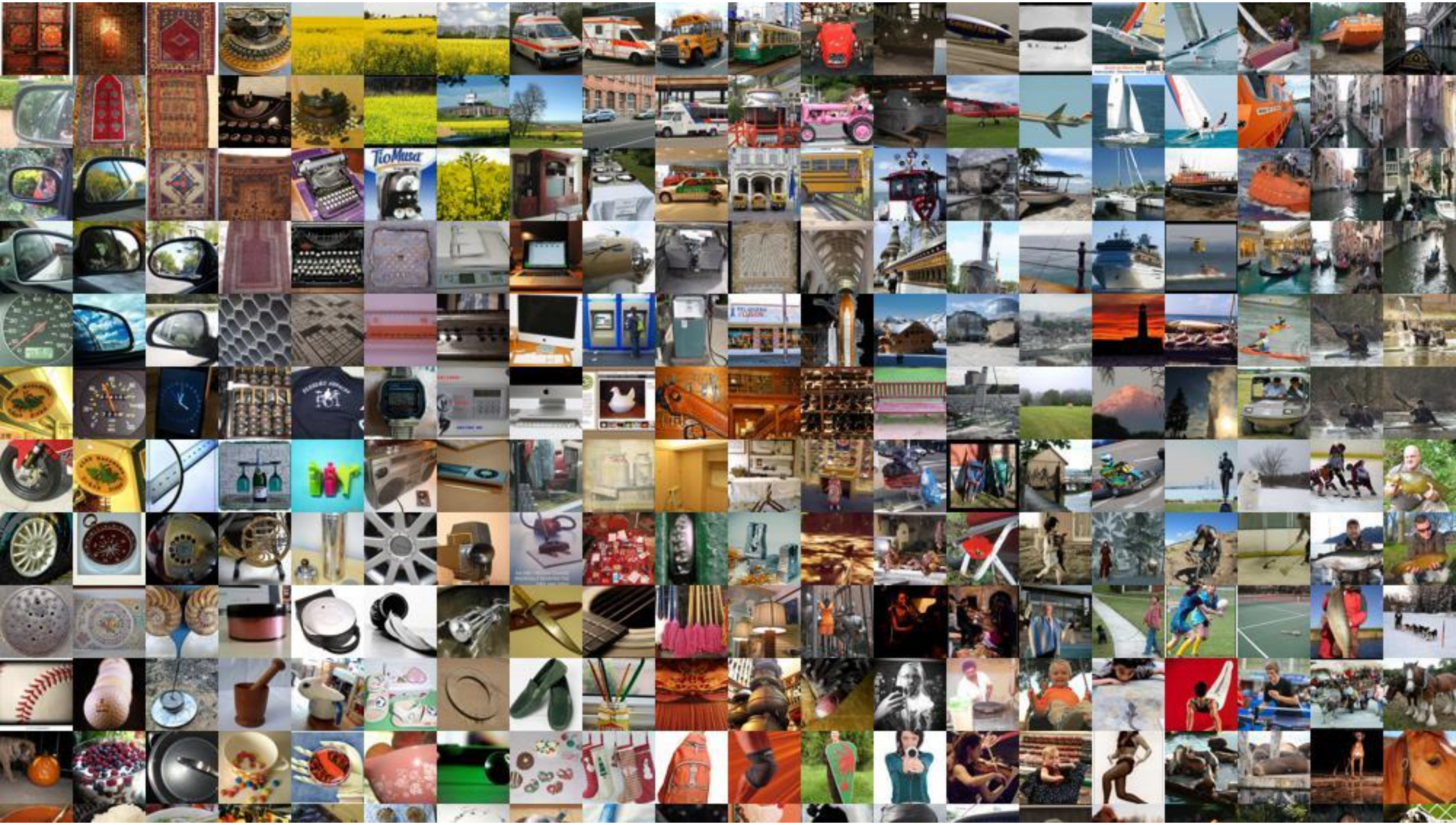
Remote sensing is becoming cheaper and more accurate

Satellite Imagery

Deep Learning for Image Classification



Challenge: Little Ground Truth Data for Poverty



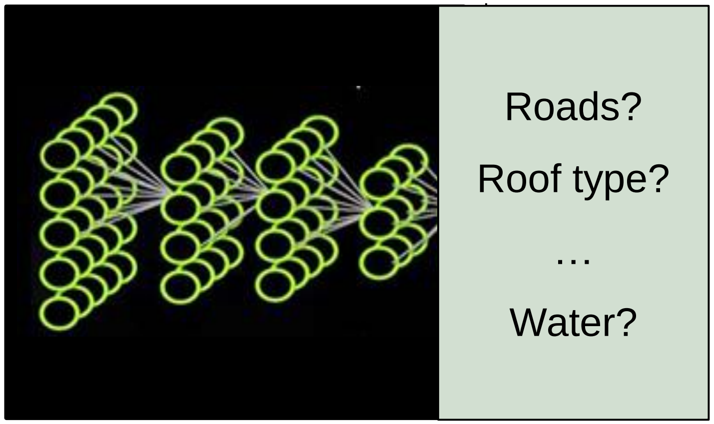
Extracting Socioeconomic Data from High-resolution Daytime Satellite Imagery

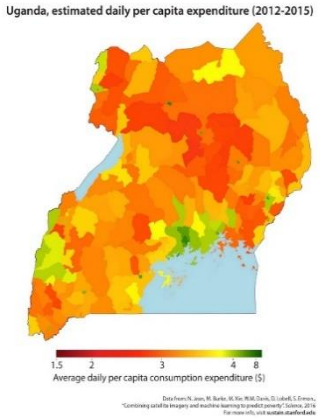
Input:

images

Output:

poverty estimates





Convolutional Neural Network

+

Transfer Learning

+

Semi-supervised Learning

Xie et a, AAAI-16

Jean et al, *Science*, 2016

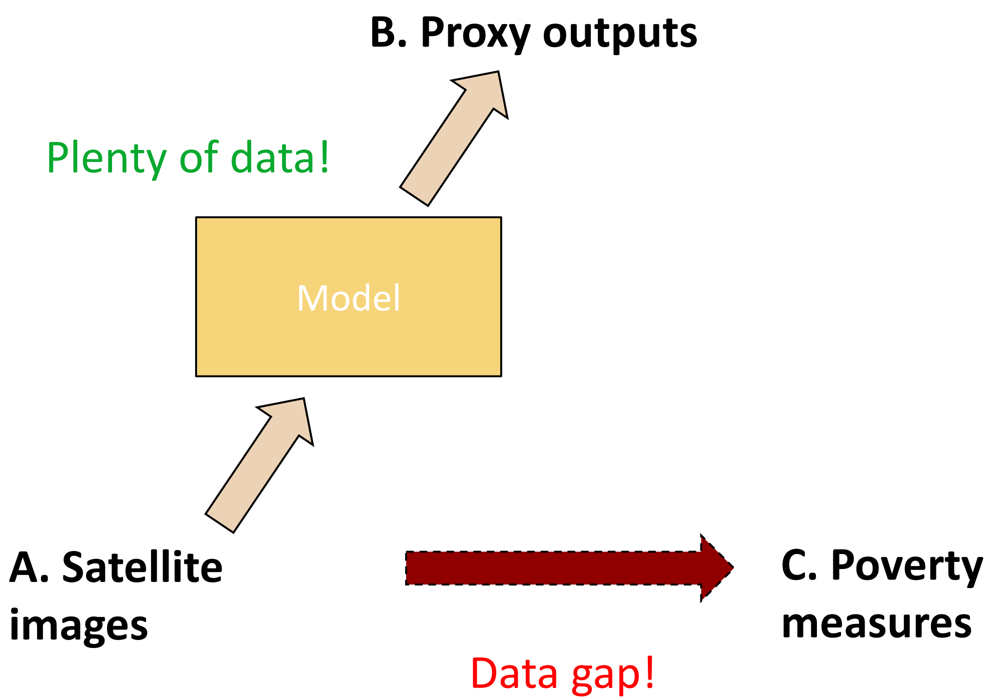
Transfer learning bridges the data gap

A. Satellite images

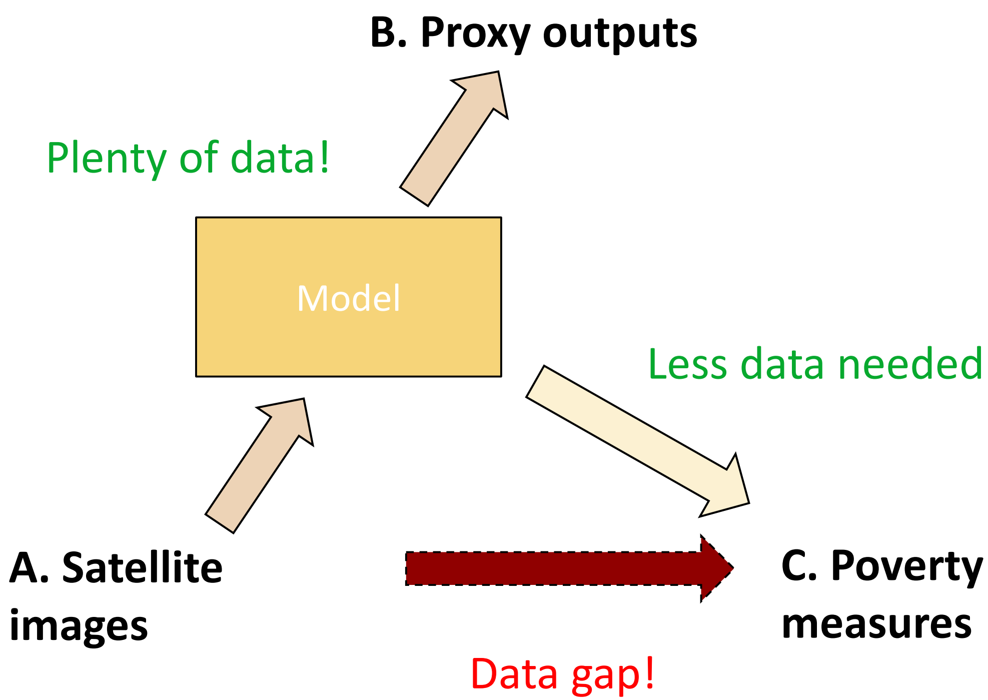
Data gap!

C. Poverty measures

Transfer learning bridges the data gap



Transfer learning bridges the data gap



Nighttime lights as proxy for economic development

Nighttime lights as proxy for economic development

Nighttime lights as proxy for economic development

Satellite Imagery → Proxy

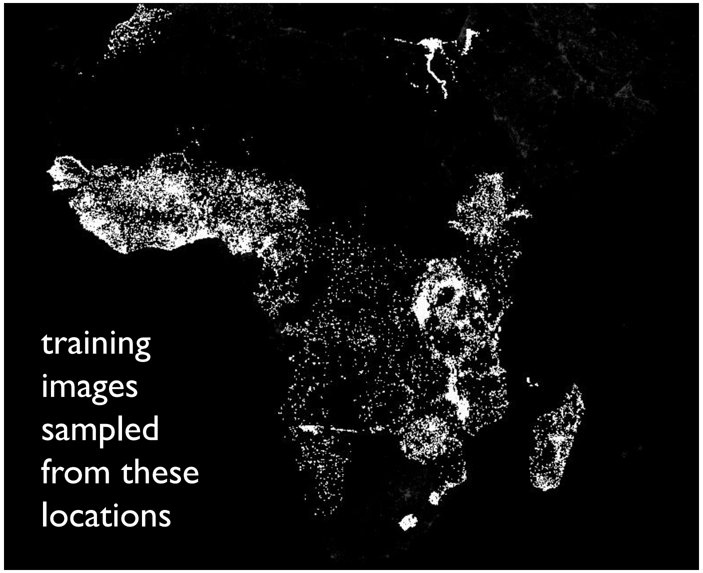
Training data on the proxy task is plentiful

Labeled input/output training pairs

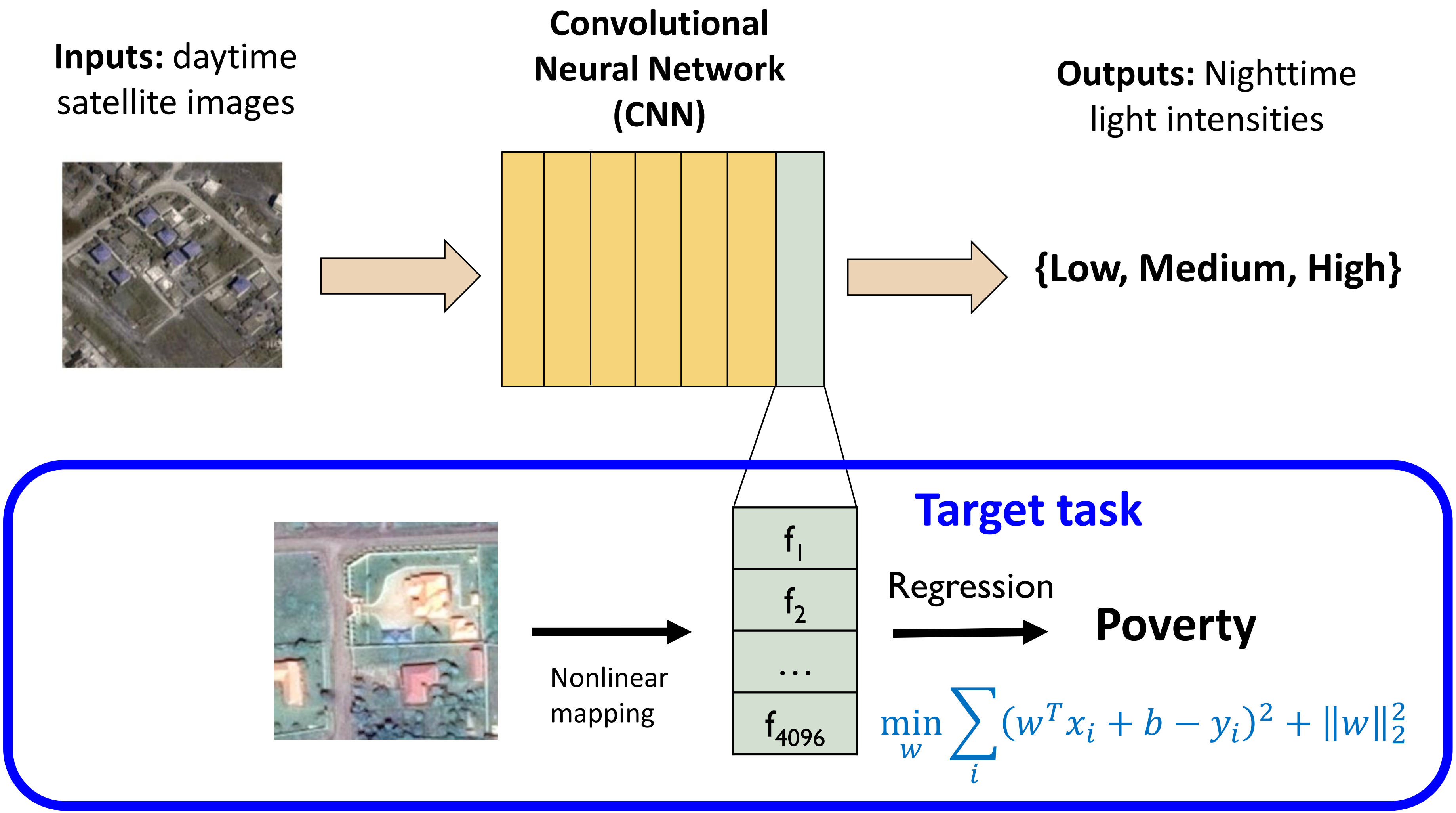
( , Low nightlight intensity)

…

( , High nightlight intensity)



Satellite Imagery → Proxy → Poverty

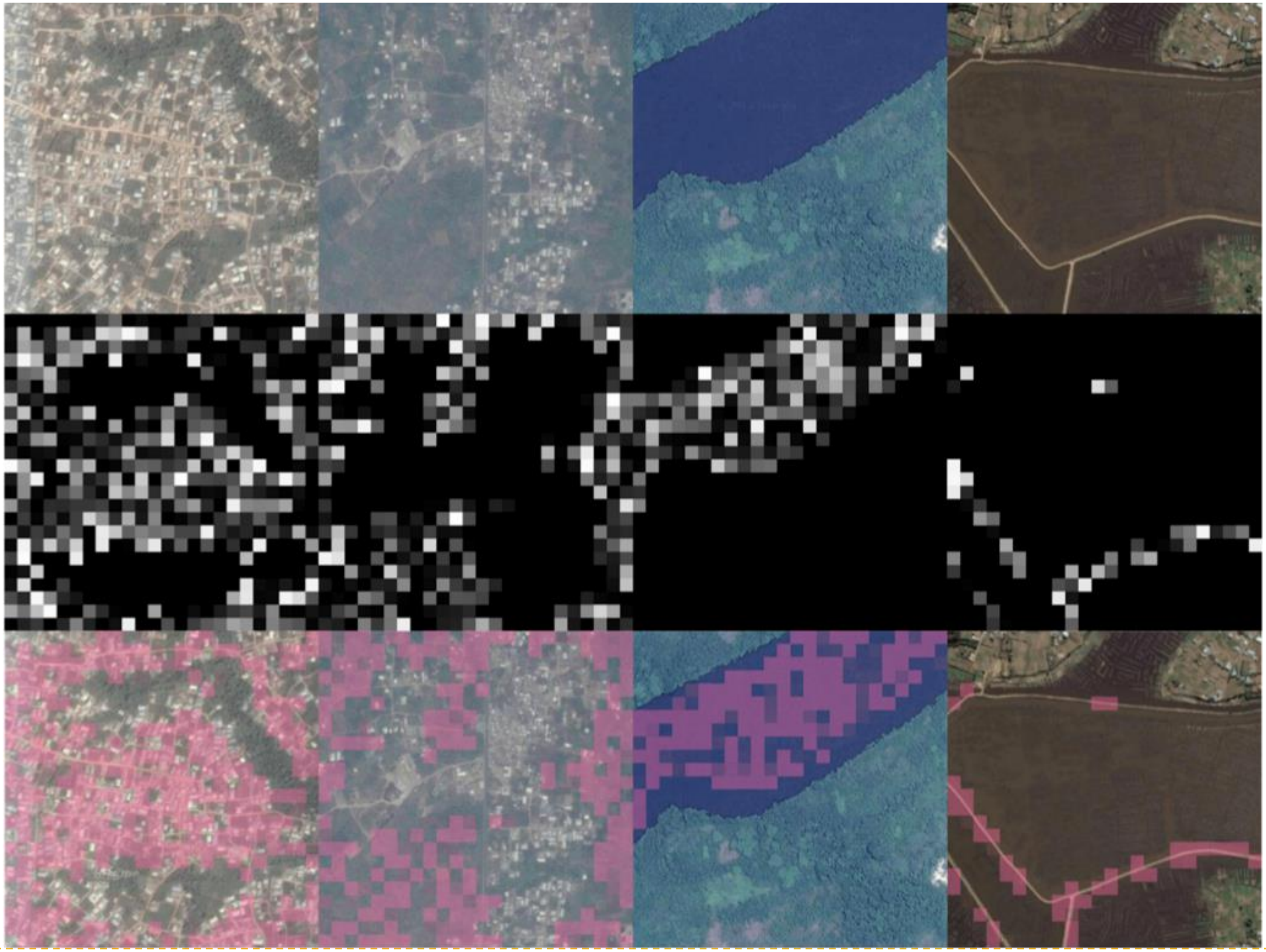


Satellite Imagery → Poverty

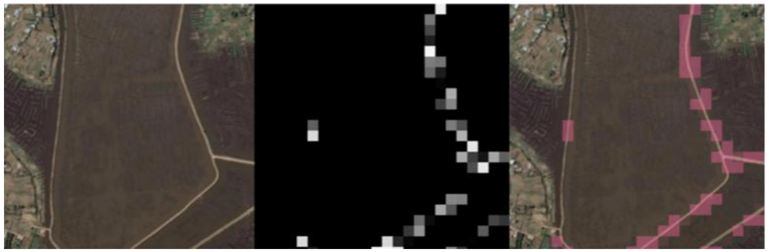
- Estimate either average household expenditures or average household wealth at the “cluster” level
- Transfer Learning Pipeline

CNN	Image Classification: identify low level image features: edges and corners
Fine-tuned CNN	Feature Extraction: Predict the nighttime light intensities
Ridge Regression	Estimate cluster-level expenditure or assets

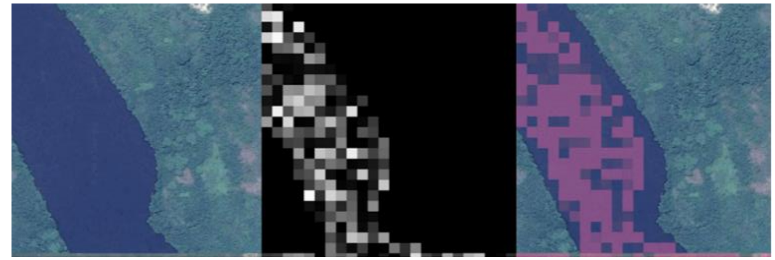
Satellite Imagery → Poverty



Satellite Imagery → Poverty

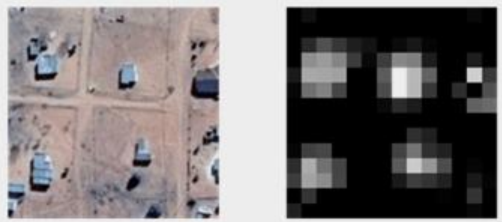


Roads

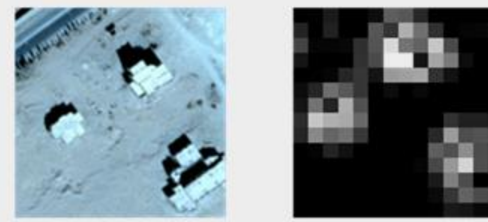


Water

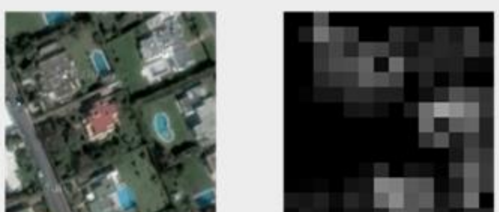
Traffic



Houses



Tall buildings

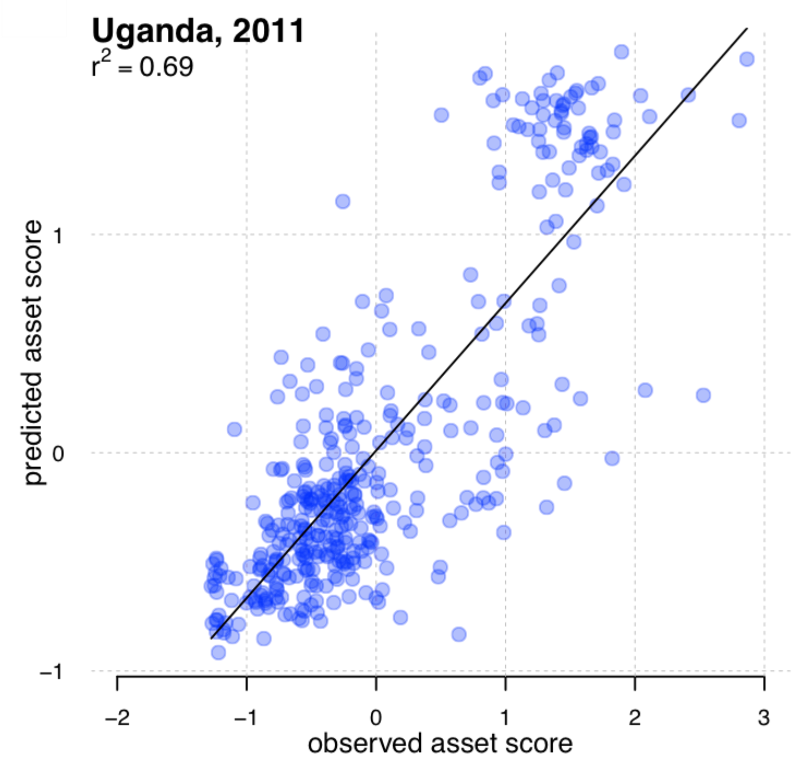


Swimming Pools

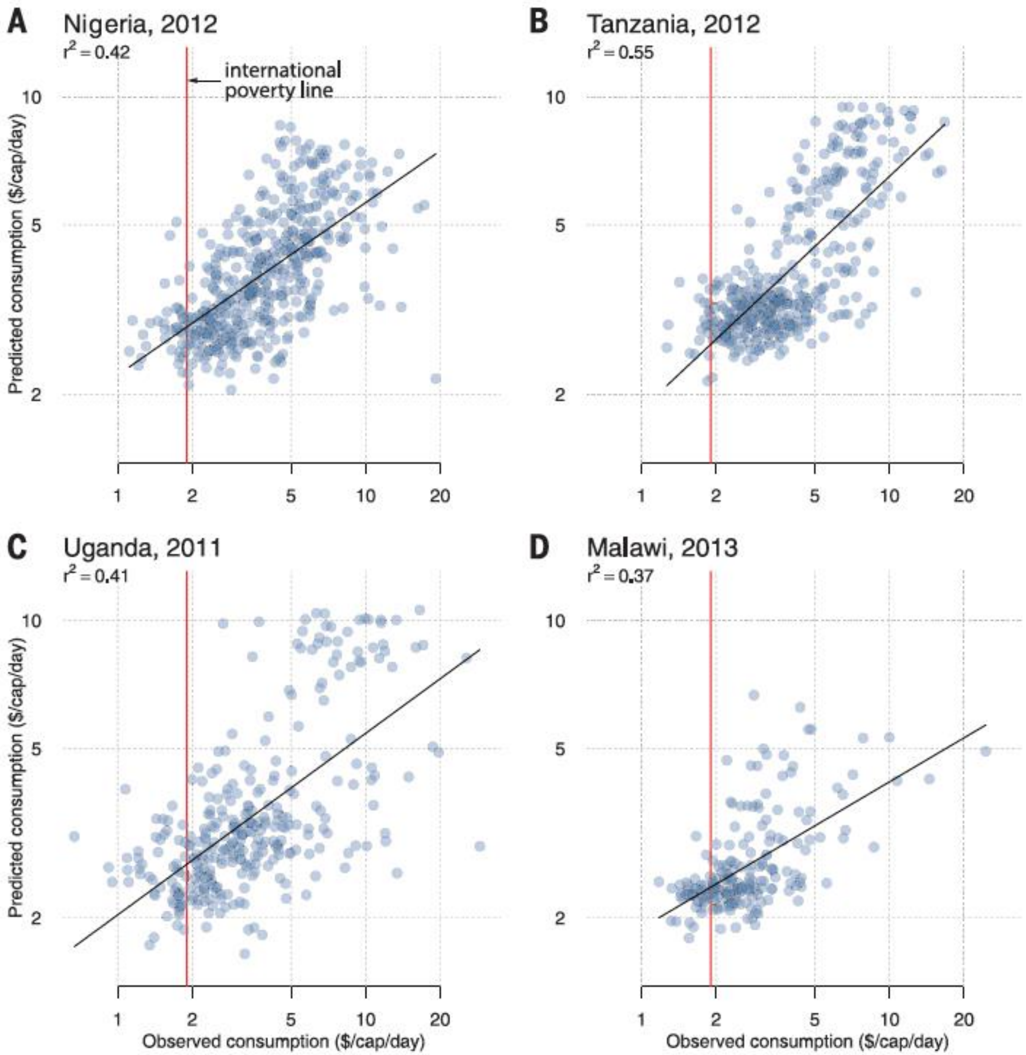
Farmland

…
and thousands more

Satellite Imagery → Poverty

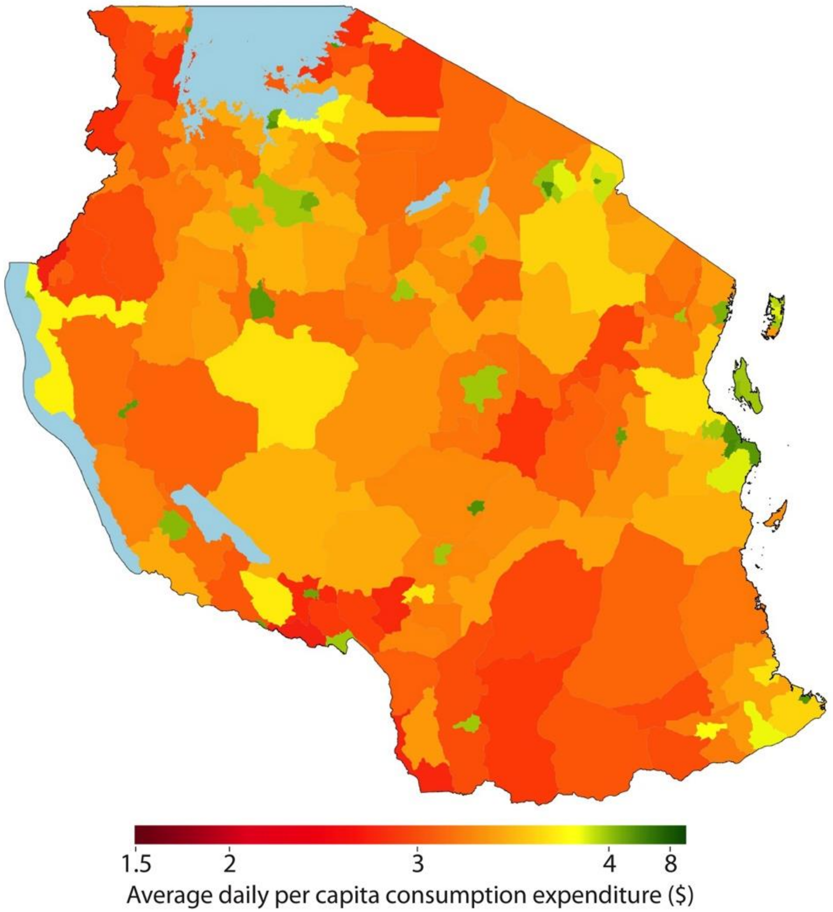


Satellite Imagery → Poverty

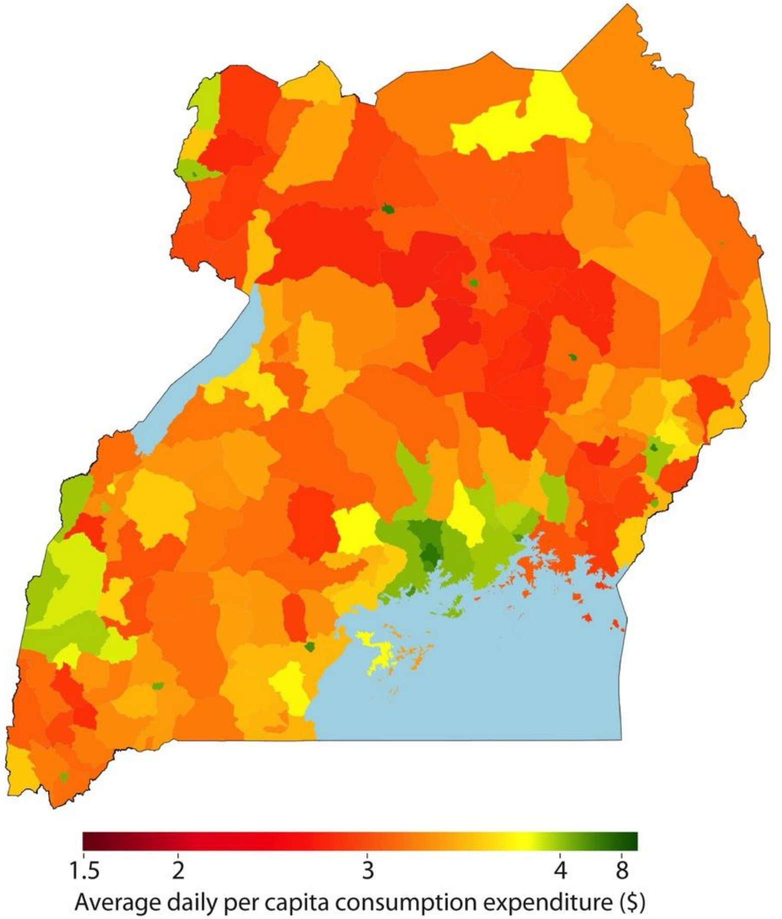


Satellite Imagery → Poverty

Tanzania, estimated daily per capita expenditure (2012-2015)

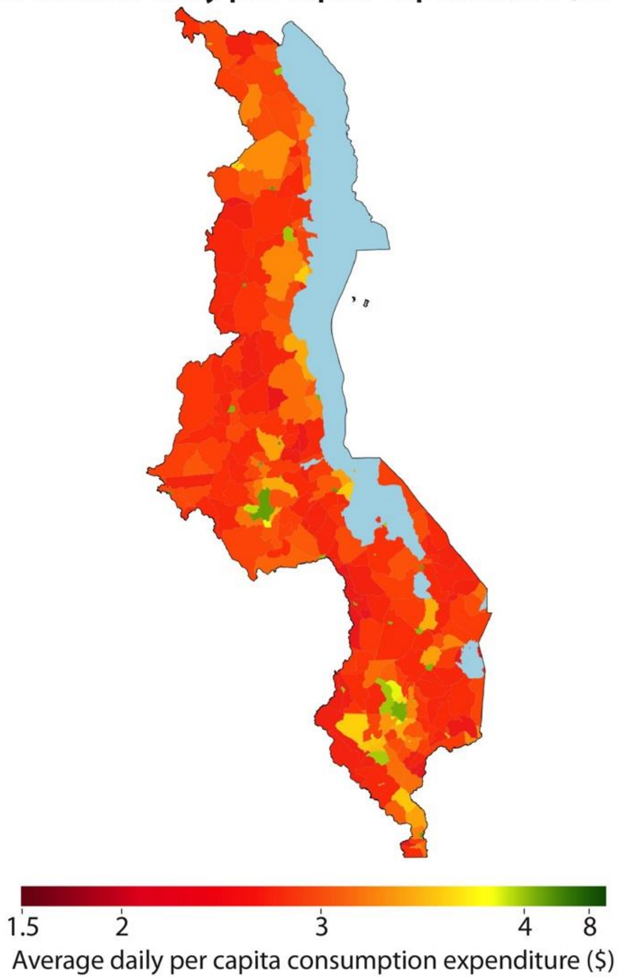


Uganda, estimated daily per capita expenditure (2012-2015)

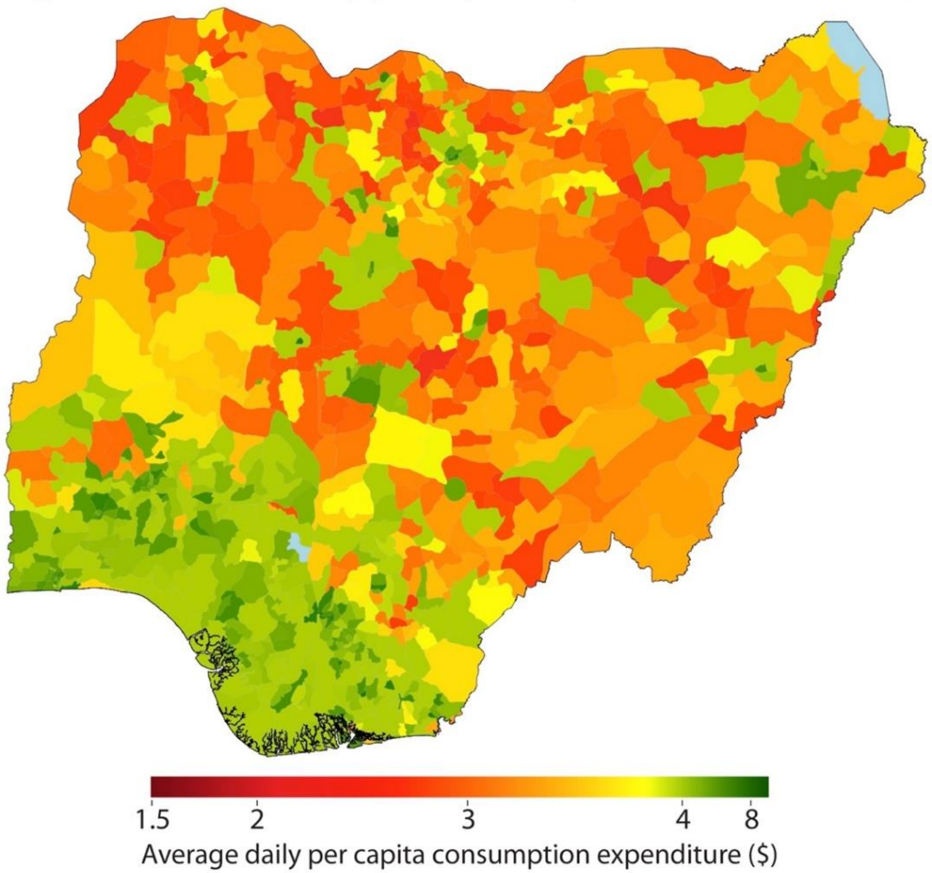


Satellite Imagery → Poverty

Malawi, estimated daily per capita expenditure (2012-2015)



Nigeria, estimated daily per capita expenditure (2012-2015)



Impact


Bill Gates
@BillGates
Follow
This imaging technique could make it easier fo
aid to reach the people who need it the most.
How satellite images are helping find the world's hidden poor
Images from space hold a secret to helping some of the world's poorest people


DISEASE FACTORIES
Why the government won't stop the problem at pig farms
What our stories say about our prehistoric past
THE EVOLUTION OF MYTHS
BIRTH OF OUR SOLAR SYSTEM
A (surprisingly violent) blur of collisions
Is carbon from permafrost speeding up climate change?
AS THE TUNDRA THAWS
SCIENTIFIC AMERICAN
10 IDEAS
THAT WILL CHANGE THE WORLD
INSIDE
A BATTERY THAT EATS CARBON
CLOTHES THAT COOL
PREDICTING POVERTY
THE (UNHACKABLE) QUANTUM INTERNET
SUPERMATERIALS FROM SUPERATOMS
A SHIELD AGAINST VIRUSES
ANTIBIOTICS FROM SCRATCH
SOFTWARE THAT CAN READ
ROBOT IN A PILL
BLOOD TESTS AS CHEAP AS PAPER

References

- [[1] Combining satellite imagery and machine learning to predict poverty]
- [[3] (video) Measuring progress towards sustainable development goals with machine learning]

Additional Resources

- Text book
  - *[Deep Learning, Chapter 6, 9]*
  - Ian Goodfellow and Yoshua Bengio and Aaron Courville
- Online course
  - https://www.coursera.org/learn/neural-networks-deep-learning
  - https://www.coursera.org/learn/convolutional-neural-networks

Acknowledgment

- Slides on poverty estimation is adapted from slides by Prof. Stefano Ermon

Backup Slides

Jacobian Matrix

$$f: \mathbb{R}^n \to \mathbb{R}^m$$
$$x \in \mathbb{R}^n$$

$$\frac{\partial f(x)}{\partial x} =$$

When f is a scalar valued function, i.e., $f: \mathbb{R}^n \to \mathbb{R}$

$$\frac{\partial f(x)}{\partial x} =$$

Jacobian Matrix

$$f: \mathbb{R}^n \to \mathbb{R}^m$$
$$x \in \mathbb{R}^n$$

$$\frac{\partial f(x)}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

When f is a scalar valued function, i.e., $f: \mathbb{R}^n \to \mathbb{R}$

$$\frac{\partial f(x)}{\partial x} = \left(\nabla_x f(x)\right)^{\mathrm{T}}$$

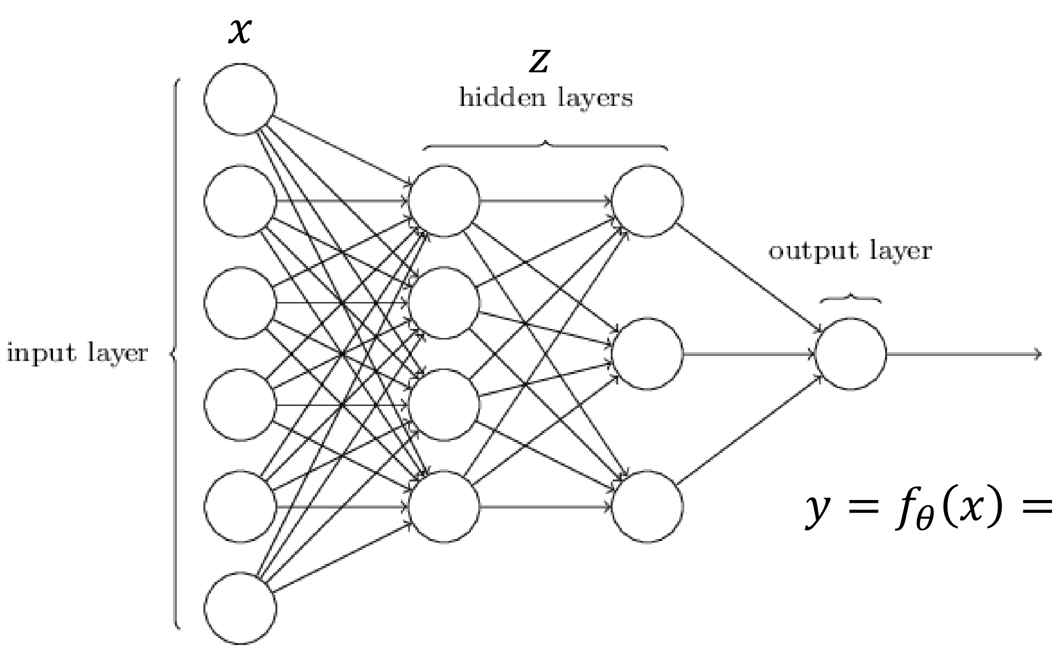
Apply Chain Rule in Backward Pass

- Let z_k denote the "input" for k^{th} layer
- Let $z_{k+1} = f^{(k)}(h_k^T z_k + b_k)$, then

$$\frac{\partial z_{k+1}}{\partial z_k} =$$

$$\frac{\partial z_{k+1}}{\partial b_k} =$$

$$\frac{\partial z_{k+1}}{\partial h_k} =$$



$$y = f_\theta(x) = f^{(K+1)}\Big(\underbrace{f^{(K)}\big(\dots f^{(2)}(\underbrace{f^{(1)}(x)}_{z_2})\big)}_{z_{K+1}}\Big)$$

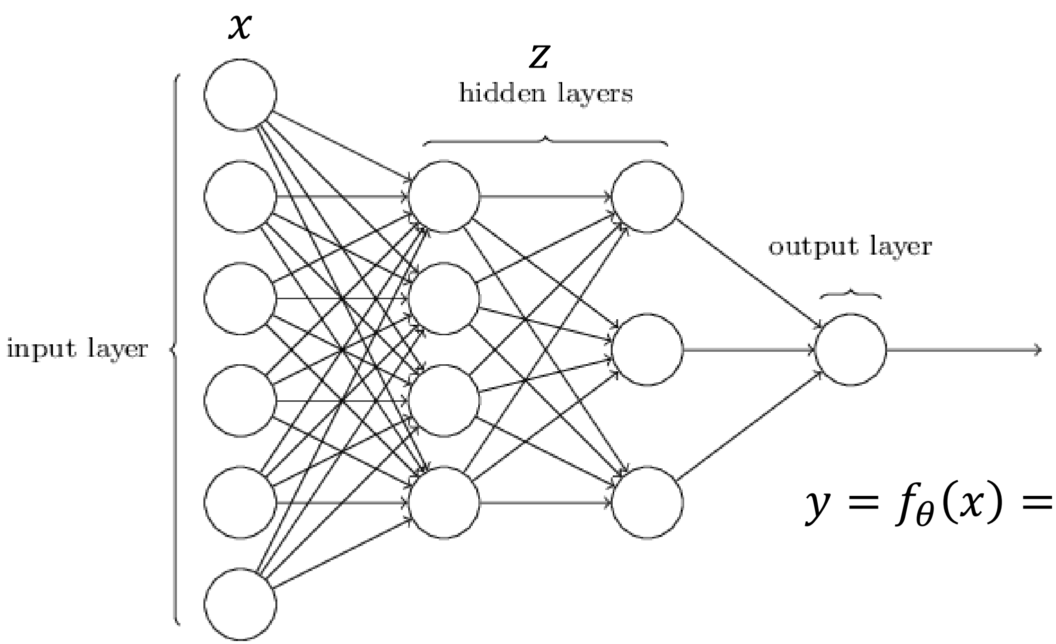
Apply Chain Rule in Backward Pass

- Let z_k denote the "input" for k^{th} layer
- Let $z_{k+1} = f^{(k)}(h_k^T z_k + b_k)$, then

$$\frac{\partial z_{k+1}}{\partial z_k} = \frac{\partial f^{(k)}\left(h_k^T z_k + b_k\right)}{\partial\left(h_k^T z_k + b_k\right)} h_k$$

$$\frac{\partial z_{k+1}}{\partial b_k} = \frac{\partial f^{(k)}\left(h_k^T z_k + b_k\right)}{\partial\left(h_k^T z_k + b_k\right)}$$

$$\frac{\partial z_{k+1}}{\partial h_k} = \frac{\partial f^{(k)}\left(h_k^T z_k + b_k\right)}{\partial\left(h_k^T z_k + b_k\right)} z_k^T$$



$$y = f_\theta(x) = f^{(K+1)}\Bigg(\underbrace{f^{(K)}\Big(\ldots f^{(2)}\big(\underbrace{f^{(1)}(x)}_{z_2}\big)\Big)}_{z_{K+1}}\Bigg)$$

Back Propagation

$$\frac{\partial l(f_\theta(x), \hat{y})}{\partial \theta_j} = \frac{\partial l(y, \hat{y})}{\partial y} \frac{\partial y}{\partial z_K} \frac{\partial z_K}{\partial z_{K-1}} \cdots \frac{\partial z_{j+1}}{\partial \theta_j}$$

Back Propagation

$z_0 \leftarrow x$

For $k = 1..K$

Compute $z_{k+1} =$

Compute $z'_{k+1} =$

$L \leftarrow l(z_{K+1}, \hat{y})$

Compute $g_{K+1} =$

For $k = K..1$

Compute $g_k =$

$\nabla_{b_k} \leftarrow$

$\nabla_{h_k} \leftarrow$

$$\frac{\partial z_{k+1}}{\partial z_k} = \frac{\partial f^{(k)}\left(h_k^T z_k + b_k\right)}{\partial (h_k^T z_k + b_k)} h_k$$

$$\frac{\partial z_{k+1}}{\partial b_k} = \frac{\partial f^{(k)}\left(h_k^T z_k + b_k\right)}{\partial (h_k^T z_k + b_k)}$$

$$\frac{\partial z_{k+1}}{\partial h_k} = \frac{\partial f^{(k)}\left(h_k^T z_k + b_k\right)}{\partial (h_k^T z_k + b_k)} z_k^T$$

Back Propagation

$$\frac{\partial l(f_\theta(x),\hat{y})}{\partial \theta_j} = \frac{\partial l(y,\hat{y})}{\partial y}\frac{\partial y}{\partial z_K}\frac{\partial z_K}{\partial z_{K-1}} \cdots \frac{\partial z_{j+1}}{\partial \theta_j}$$

Back Propagation

$z_0 \leftarrow x$

For $k = 1..K$

Compute $z_{k+1} = f^{(k)}(h_k^T z_k + b_k)$

Compute $z'_{k+1} = f^{(k)\prime}(h_k^T z_k + b_k)$

$L \leftarrow l(z_{K+1}, \hat{y})$

Compute $g_{K+1} = \frac{\partial l(z_{K+1},\hat{y})}{\partial z_{K+1}}$

For $k = K..1$

Compute $g_k = h_k^T(g_{k+1} \circ z'_{k+1})$

$\nabla_{b_k} \leftarrow g_{k+1} \circ z'_{k+1}$

$\nabla_{h_k} \leftarrow (g_{k+1} \circ z'_{k+1}) z_k^T$

$$z'_{k+1} = \frac{\partial f^{(k)}(h_k^T z_k + b_k)}{\partial (h_k^T z_k + b_k)}$$

Dot product

$$\frac{\partial z_{k+1}}{\partial z_k} = \frac{\partial f^{(k)}(h_k^T z_k + b_k)}{\partial (h_k^T z_k + b_k)} h_k$$

$$\frac{\partial z_{k+1}}{\partial b_k} = \frac{\partial f^{(k)}(h_k^T z_k + b_k)}{\partial (h_k^T z_k + b_k)}$$

$$\frac{\partial z_{k+1}}{\partial h_k} = \frac{\partial f^{(k)}(h_k^T z_k + b_k)}{\partial (h_k^T z_k + b_k)} z_k^T$$

$$g_k = \frac{\partial l}{\partial z_k} = \frac{\partial l}{\partial z_{k+1}}\frac{\partial z_{k+1}}{\partial z_k} = h_k^T(g_{k+1} \circ z'_{k+1})$$