(Annen) BergNet

Linda Zeng
Harvard University
Pre-College Program

Presented on August 2, 2024

How can we train a neural network to analyze and recognize Annenberg Hall's food?

Motivation

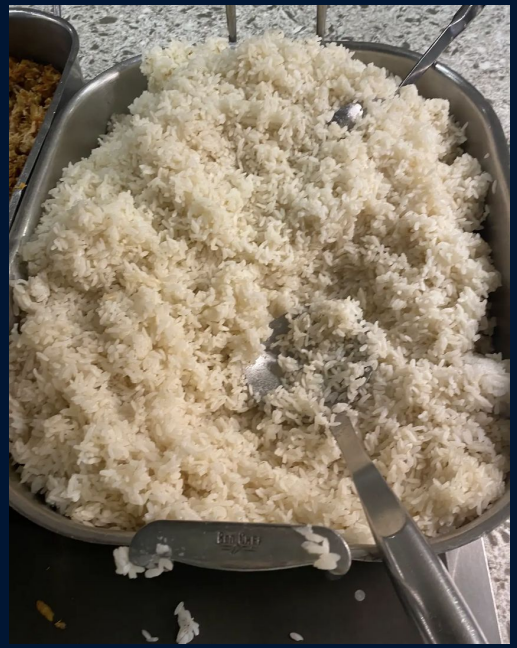
This Class

- Ubiquitous part of everyone **here**'s life
- Close to us (literally)
- Fun idea

In General

- Nutrition recommendation
- Balancing meals
- Monitor food products for food processing
- Machine understanding of images in our world

Task



224 x 224

Classify

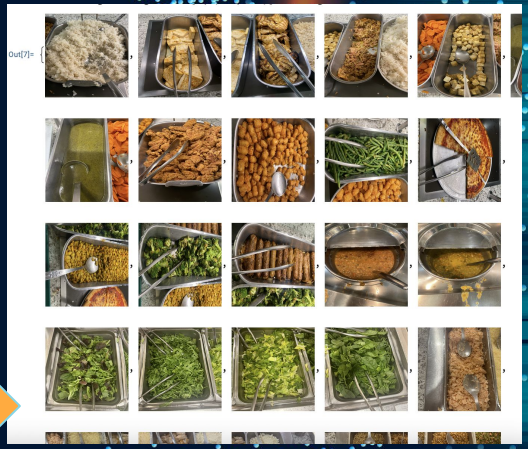
Protein
Grains
Vegetable
Fruit
Soup
Other

Data Collection

Data Labeling

	A	B	
1	Path	Category	Name
2	IMG_3460.HEIC	Grains	Rice
3	IMG_3461.HEIC	Protein	Tofu
4	IMG_3462.HEIC	Protein	Chicken
5	IMG_3463.HEIC	Protein	Pulled Chicken
6	IMG_3464.HEIC	Protein	Tofu
7	IMG_3465.HEIC	Vegetable	Carrot
8	IMG_3466.HEIC	Soup	Green Curry
9	IMG_3467.HEIC	Protein	Fried Chicken
10	IMG_3468.HEIC	Grains	Tater Tots
11	IMG_3469.HEIC	Vegetable	Green Beans
12	IMG_3470.HEIC	Other	Pizza
13	IMG_3471.HEIC	Vegetable	Lima Beans
14	IMG_3472.HEIC	Vegetable	Broccoli
15	IMG_3473.HEIC	Protein	Sausage
16	IMG_3474.HEIC	Soup	Tomato Basil Soup
17	IMG_3475.HEIC	Soup	Chicken Tortilla Soup
18	IMG_6164.HEIC	Vegetable	Spring Mix
19	IMG_6165.HEIC	Vegetable	Baby Arugula
20	IMG_6166.HEIC	Vegetable	Lettuce
21	IMG_6167.HEIC	Vegetable	Spinach
22	IMG_6168.HEIC	Protein	Tuna

Data Loading



(Surprising) Challenge

How to add images to dataset? → Transpose!

```
In[] := newMatrix = Transpose[Join[Transpose[data], {resizedimages}]]
Out[]= {{IMG_3460.HEIC, Grains, Rice,  },
 {IMG_3461.HEIC, Protein, Tofu,  }, {IMG_3462.HEIC, Protein, Chicken, 
 {IMG_3463.HEIC, Protein, Pulled Chicken,  },
 {IMG_3464.HEIC, Protein, Tofu,  }, {IMG_3465.HEIC, Vegetable, Carrot, 
```

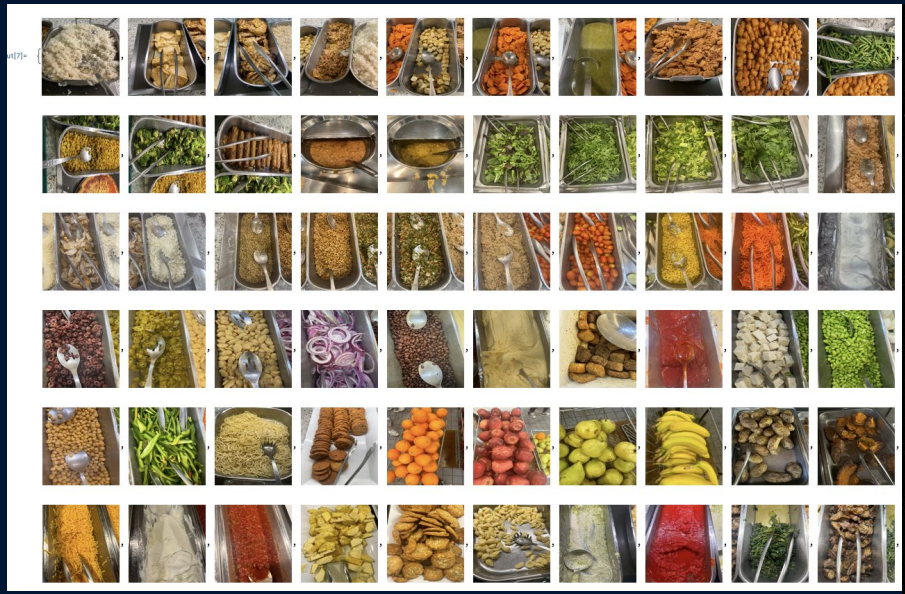
Data Scarcity Problem

67 Samples of 224 x 224 pixels.

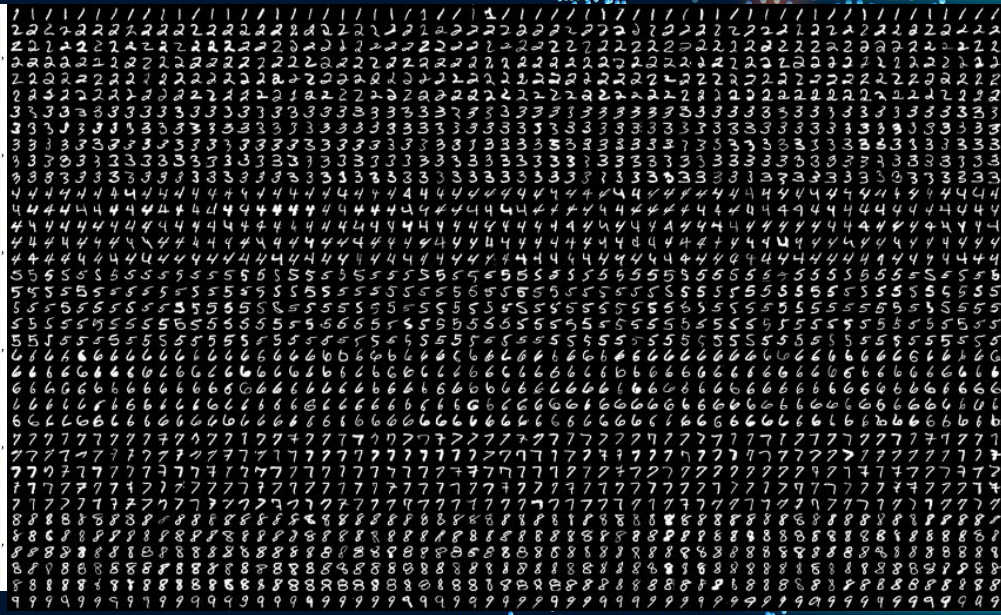
45 training samples

22 test samples

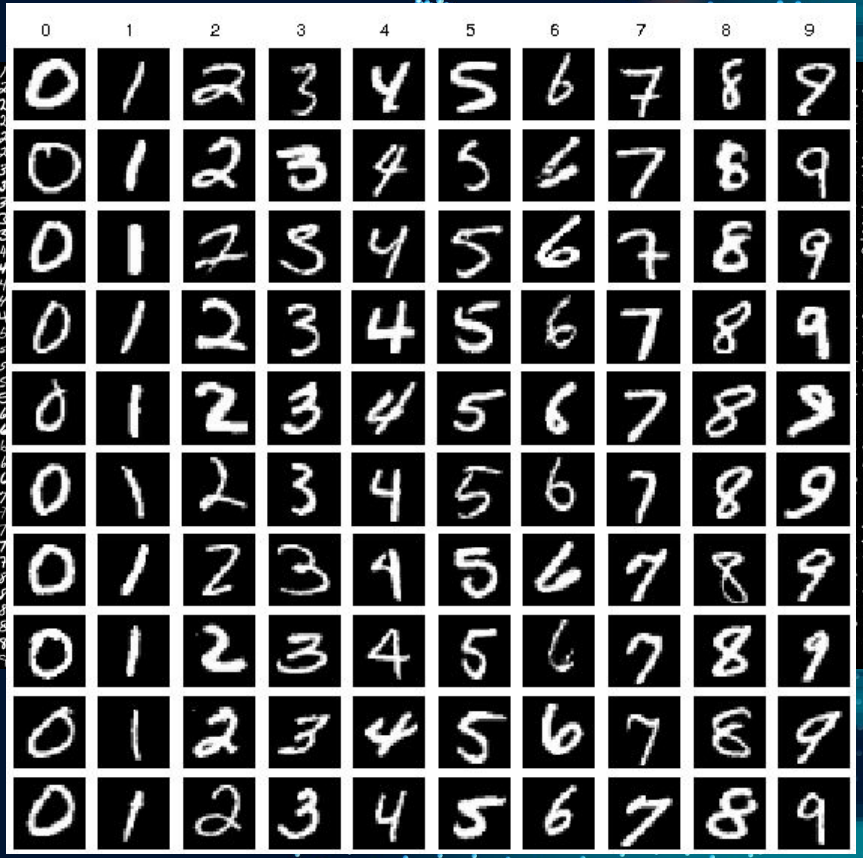
```
In[6]:= Length[images]

Out[6]= 67
```



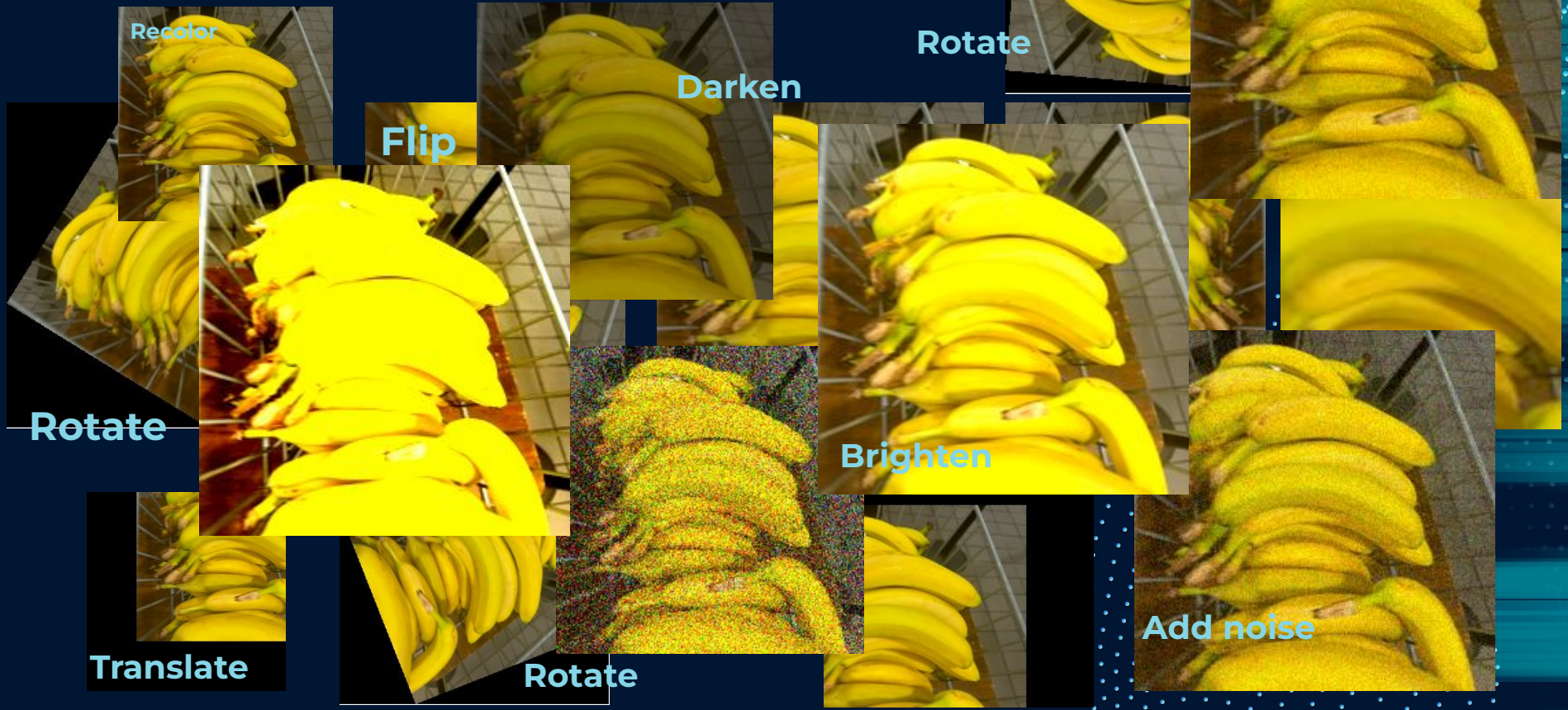
VS



VS



Fake it til you make it!



Data Scarcity Problem

67 Samples of 224 x 224 pixels.

45 training samples

22 test samples

```
In[6]:= Length[images]

Out[6]= 67
```

Data Scarcity Problem

```
In[6]:= Length[images]

Out[6]= 67
```

67 Samples of 224 x 224 pixels.

45 training samples

22 test samples

x 17

765 training samples

Training

		image
	Input	array (size: 3×224×224)
1	ConvolutionLayer	array (size: 16×222×222)
2	Ramp	array (size: 16×222×222)
3	PoolingLayer	array (size: 16×111×111)
4	ConvolutionLayer	array (size: 32×109×109)
5	Ramp	array (size: 32×109×109)
6	PoolingLayer	array (size: 32×54×54)
7	FlattenLayer	vector (size: 93312)
8	LinearLayer	vector (size: 128)
9	Ramp	vector (size: 128)
10	LinearLayer	vector (size: 6)
11	SoftmaxLayer	vector (size: 6)
	Output	class

Training

		image
	Input	array (size: 3×224×224)
1	ConvolutionLayer	array (size: 16×222×222)
2	Ramp	array (size: 16×222×222)
3	PoolingLayer	array (size: 16×111×111)
4	ConvolutionLayer	array (size: 32×109×109)
5	Ramp	array (size: 32×109×109)
6	PoolingLayer	array (size: 32×54×54)
7	FlattenLayer	vector (size: 93312)
8	LinearLayer	vector (size: 128)
9	Ramp	vector (size: 128)
10	LinearLayer	vector (size: 6)
11	SoftmaxLayer	vector (size: 6)
	Output	class

98.34% on training set

48.57% on test set

Overfitting

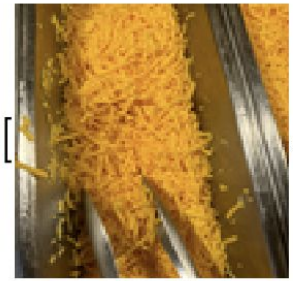
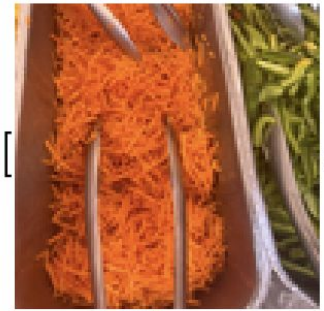
Validation loss starts going up

Training

No Dropout	Dropout 0.4	Dropout 0.1
98.34% on training set	62.68% on train	71.3% on train
48.57% on test set	**28.57%** on test	**37.14%** on test

Why?

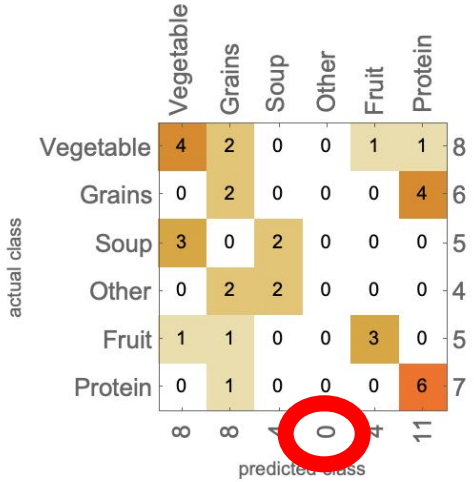
Very difficult task

```
In[121]:= trainedModel[  , {"TopProbabilities", 3}]

Out[121]= {Fruit → 0.00119143, Soup → 0.0524874, Vegetable → 0.945291}

In[122]:= trainedModel[  , {"TopProbabilities", 3}]

Out[122]= {Other → 0.00118109, Protein → 0.00547433, Vegetable → 0.991891}
```

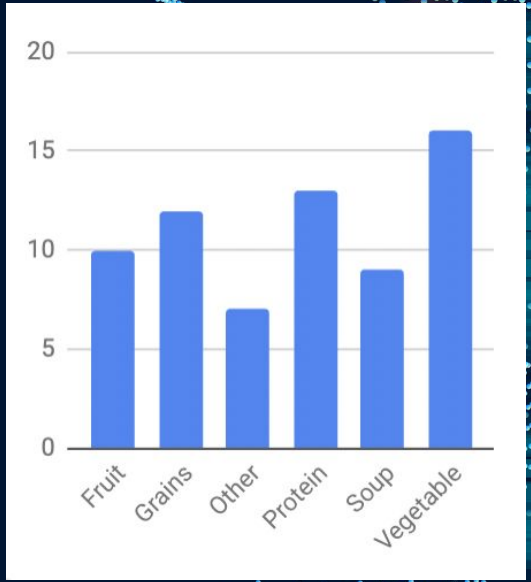
```
ClassifierMeasurements[trainedBergNet, testData]
```

Classifier Measurements

Classifier method	Net
Number of test examples	35
Accuracy	(49. ±9.)%
Accuracy baseline	(23. ±7.)%
Geometric mean of probabilities	0.118 ± 0.054
Mean cross entropy	2.14 ± 0.44
Single evaluation time	4.98 ms/example
Batch evaluation speed	330. examples/s

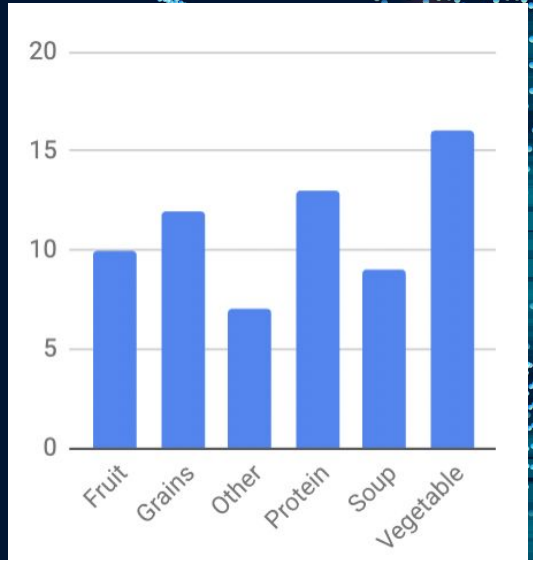


1) too little ‘Other’ samples



1) too little ‘Other’ samples

2) ‘Other’ samples are too random



1,

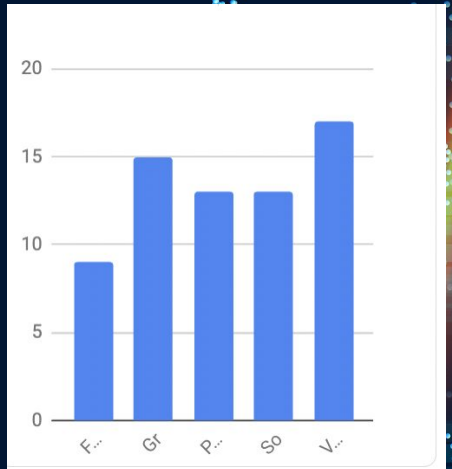
→ Other → 1,

→ Other → 1,

→ Other → 1

Removing “Other”

Going back to the **beginning**



Most | Least

VALUE	FREQUENCY
Vegetable	17
Grains	15
Soup	13
Protein	13
Fruit	9

BergNet Results

98.34% 99.34% on training set

48.57% **59% on test set**

Removed ‘Other’ class

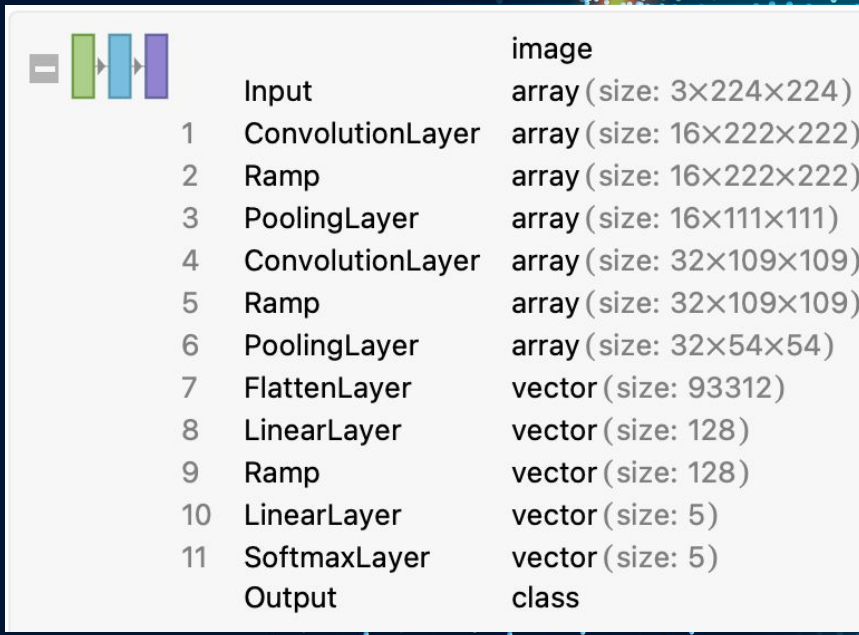
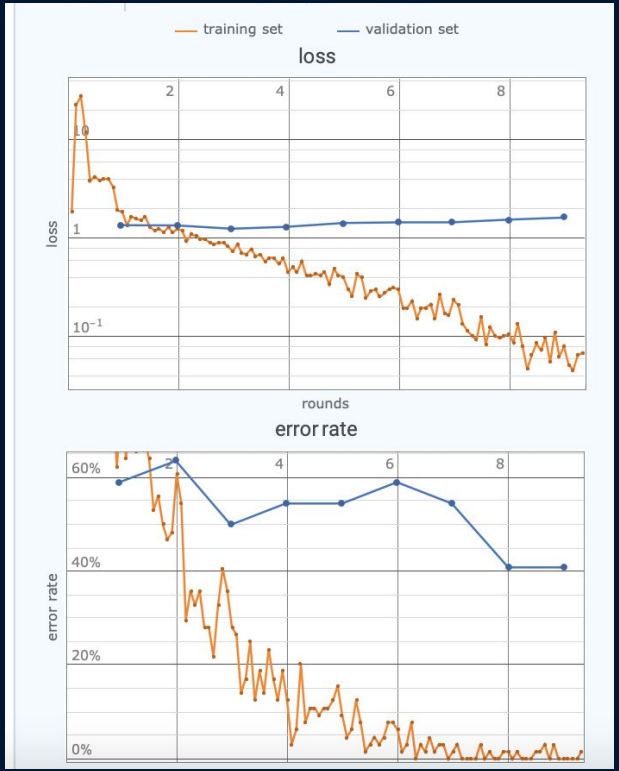
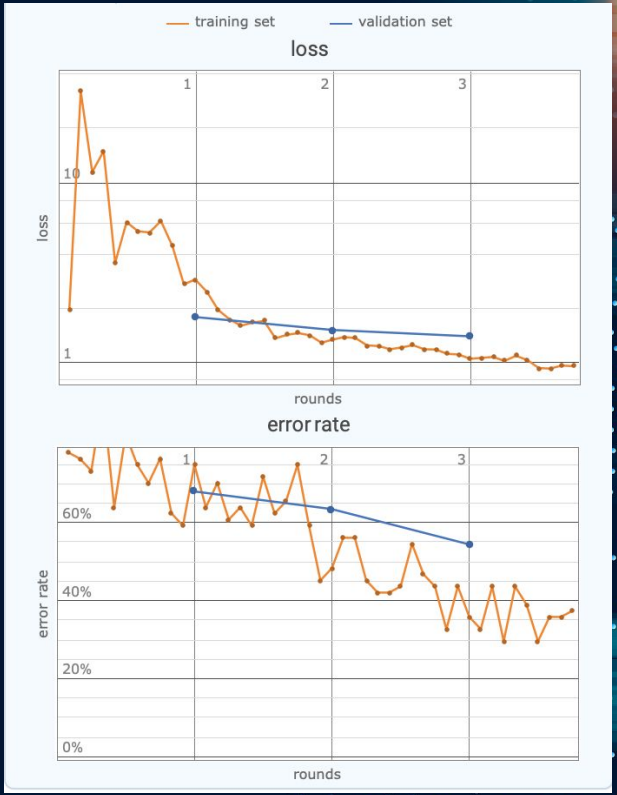


		image
	Input	array (size: 3×224×224)
1	ConvolutionLayer	array (size: 16×222×222)
2	Ramp	array (size: 16×222×222)
3	PoolingLayer	array (size: 16×111×111)
4	ConvolutionLayer	array (size: 32×109×109)
5	Ramp	array (size: 32×109×109)
6	PoolingLayer	array (size: 32×54×54)
7	FlattenLayer	vector (size: 93312)
8	LinearLayer	vector (size: 128)
9	Ramp	vector (size: 128)
10	LinearLayer	vector (size: 5)
11	SoftmaxLayer	vector (size: 5)
	Output	class

Training (trial 2)

Added dropout 0.1





BergNet Results

98.34%	99.34% on training set	91.50% on training set
48.57%	**59% on test set**	**63.63% on test set**
	Removed ‘Other’ class	Dropout 0.1

Failures

Best	**91.50%**	**63.63%**
Dropout 0.2	84.84%	59.09%
Add early stopping	77.52%	50%
Add L2 Regularization + SGD	59%	40.9%
Add L2 Regularization + Adam	93.07%	54.54%
Add Batch Norm	27%	27%

Overall results

	# Data points	Accuracy on Wednesday Test Data
No Training	0	23%
BergNet (no data augmentation)	45	50%
BergNet	765	**64%**

Examples (BergNet)

Predicted as Grains

Predicted as Protein

Predicted as Grains

Predicted as Soup

Examples

Predicted as Grains

Predicted as Grains

Predicted as Protein

Predicted as Soup

What else can we do?

- Constrained by limited data

What else can we do?

- Constrained by data

Rely on others

- There are already models trained on **similar** tasks

Fine-tune EfficientNet

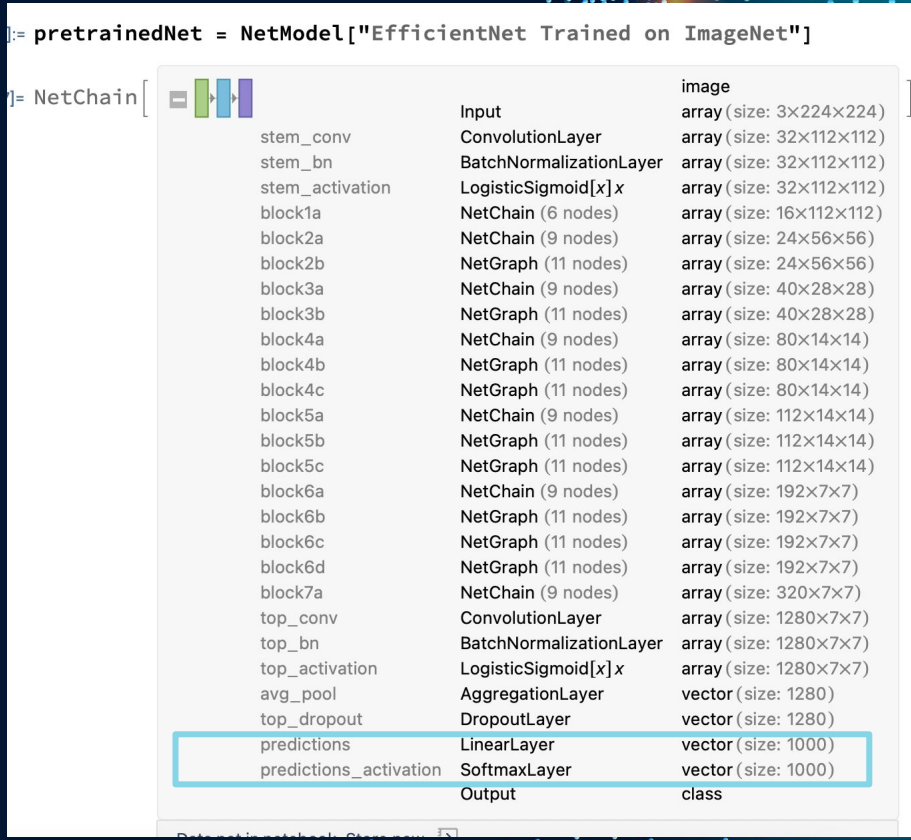
Trained to identify images

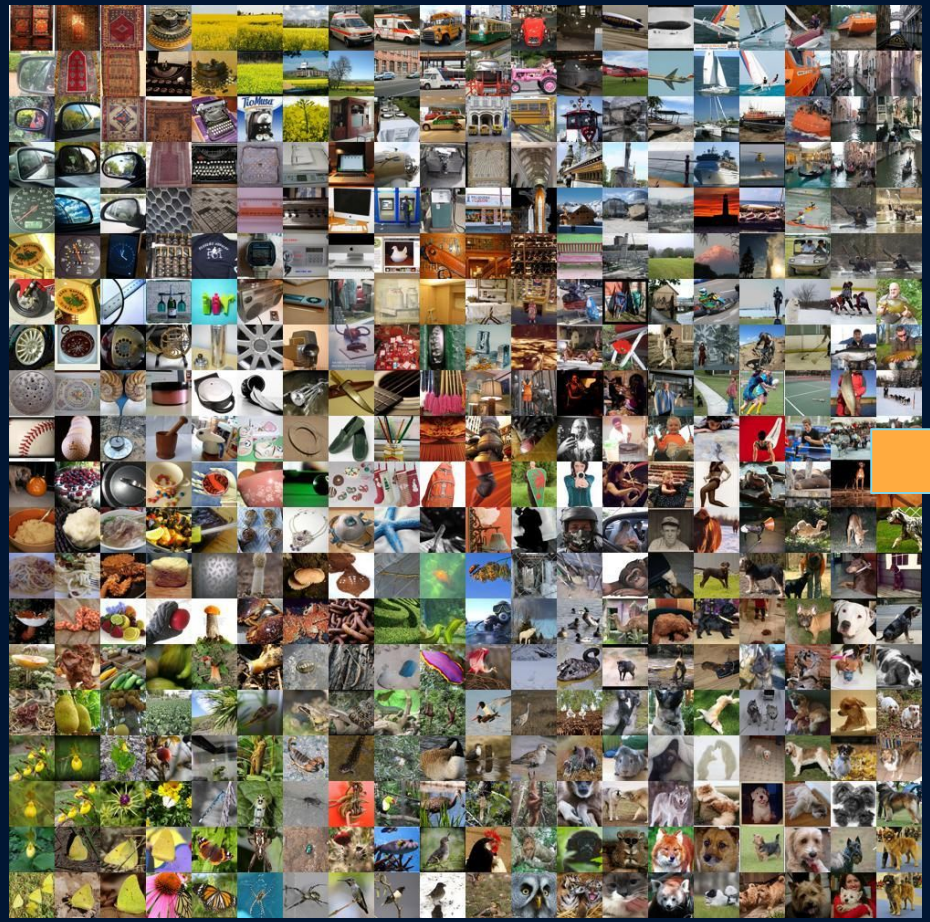
- trained on 1.2 million images
- Already optimized parameters

We can modify it for our tasks

- train on our data

Replace with 5 classes →





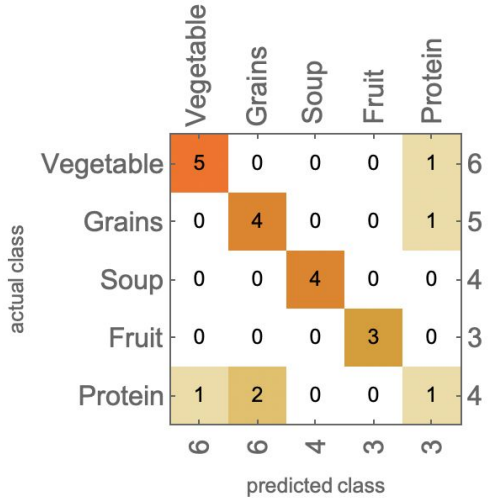
Some but not all are food pics

Add on these

Gets 77% accuracy!

Classifier Measurements

Classifier method	Net
Number of test examples	22
Accuracy	(77. ±9.)%
Accuracy baseline	(27. ±10.)%
Geometric mean of probabilities	0.326 ± 0.13
Mean cross entropy	1.12 ± 0.39
Single evaluation time	59. ms/example
Batch evaluation speed	15.1 examples/s



Overall results

	# Data points	Accuracy on Wednesday Test Data
No Training	0	23%
BergNet (no data augmentation)	45	50%
BergNet	765	**64%**

Overall results

	# Data points	Accuracy on Wednesday Test Data
No Training	0	23%
BergNet (no data augmentation)	45	50%
BergNet	765	64%
EfficientNet + ImageNet	0	14%
EfficientNet + ImageNet (Finetuned)	765	**77%**

Examples (Finetuned EfficientNet)

~~Predicted as Grains~~

Predicted as **Fruit**

Predicted as Protein

Predicted as **Protein**

~~Predicted as Grains~~

Predicted as **Protein**

~~Predicted as Soup~~

Predicted as **Vegetable**

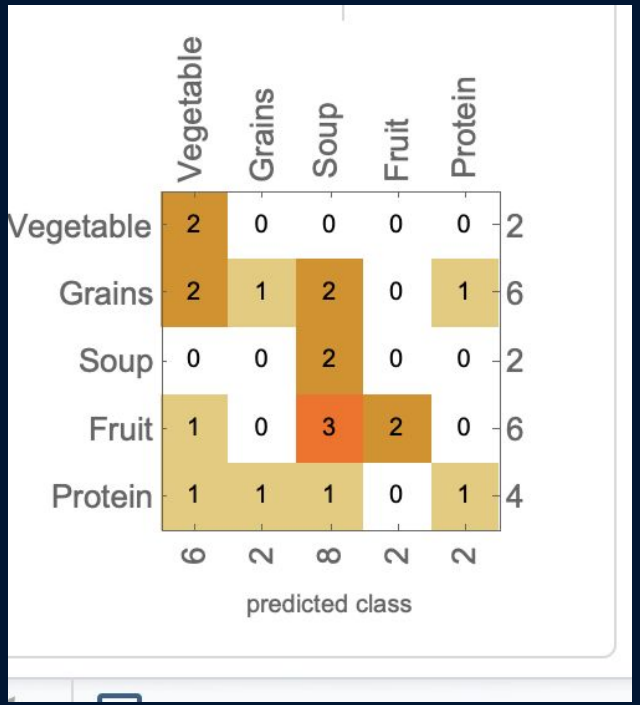
Time to test out on Thursday now!

Overall results (with Thursday)

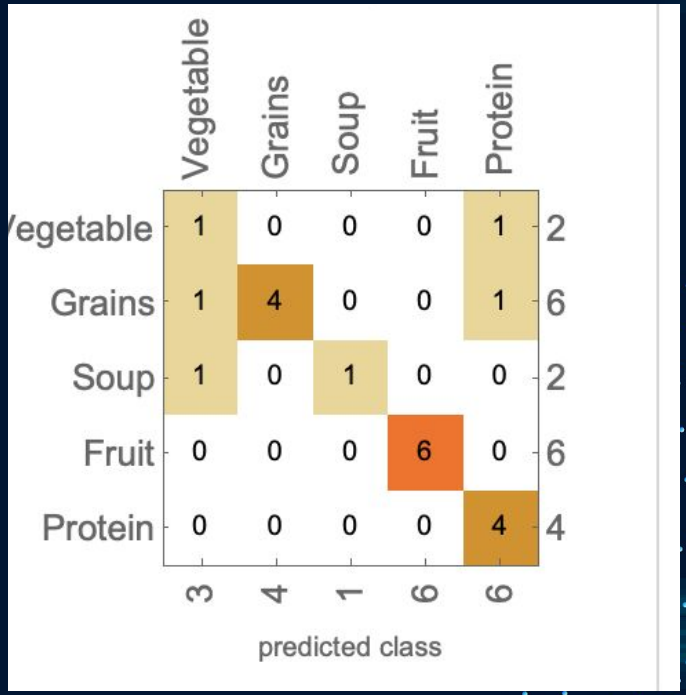
	# Data points	Accuracy on Wednesday Test Data	Accuracy on Thursday Test
No Training	0	23%	25%
BergNet (no data augmentation)	45	50%	30%
BergNet	765	64%	40%
EfficientNet + ImageNet	0	14%	35%
EfficientNet + ImageNet (finetuned)	765	**77%**	**80%**

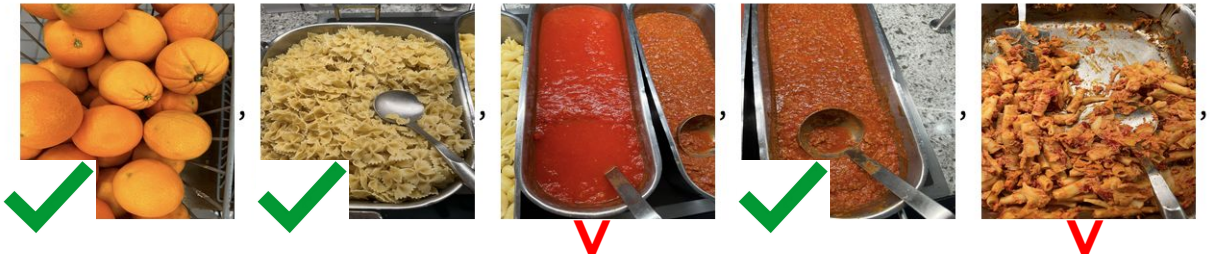
Thursday

BertNet



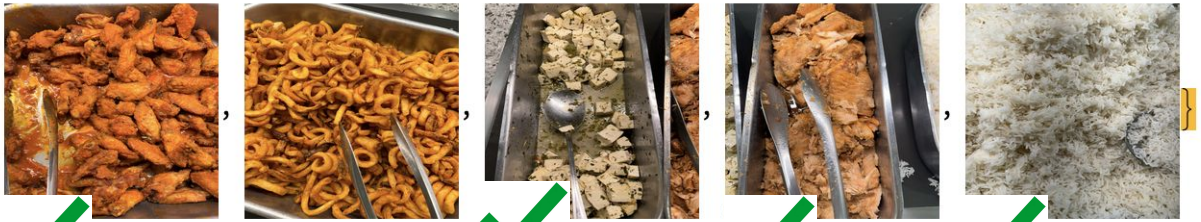
Finetuned EfficientNet





V V

P



V

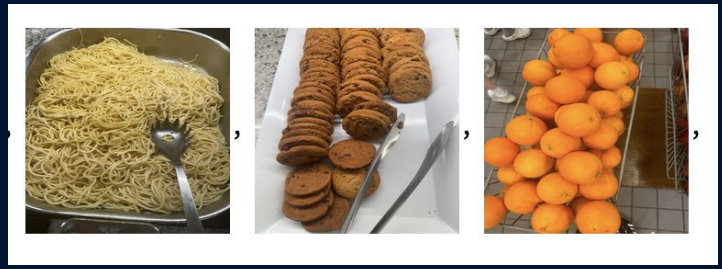
P

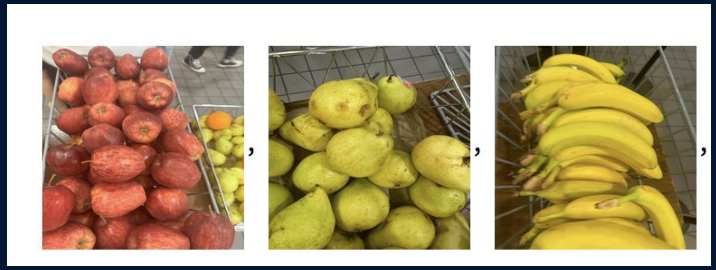
V

V

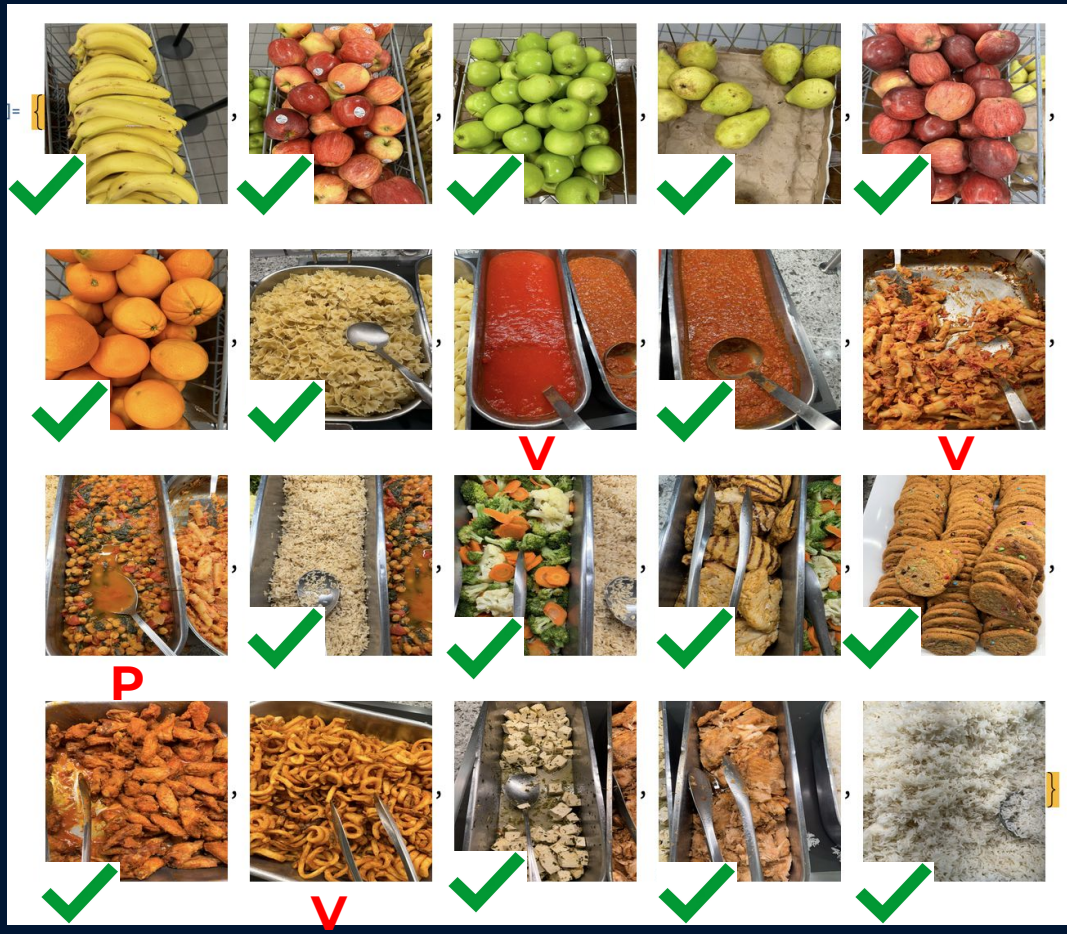
V

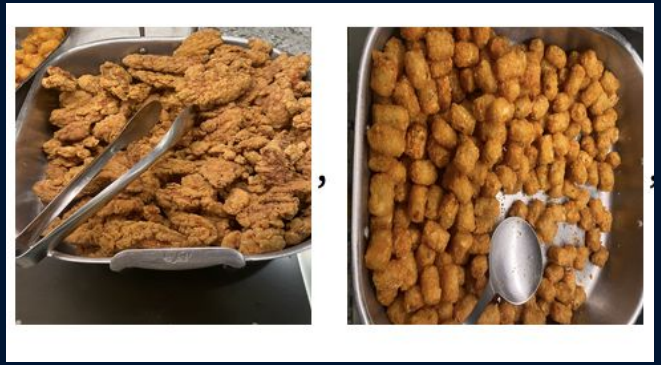
Everyday foods



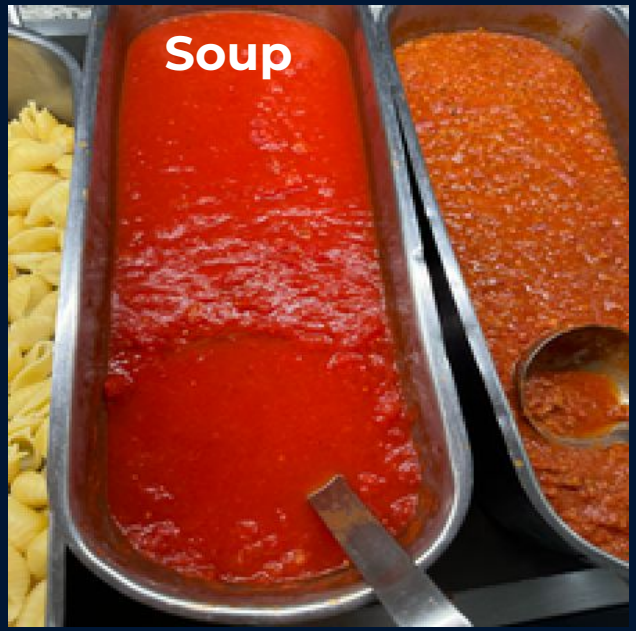


Exceptions





Exceptions



Grains -> 0.106108
Soup -> 0.375381
Vegetable -> 0.47878

Grains -> 0.349892
Soup -> 0.412876
Vegetable -> 0.148721

Conclusion

- We successfully got 77% on Wednesday and 80% on Thursday
- There are ways to work with low amounts of data
    - **Fake it til you make it**: Data augmentation
    - **Rely on others:** Fine tuning
- The more consistent food Annenberg makes, the better BergNet performs