Apparel Image Classification

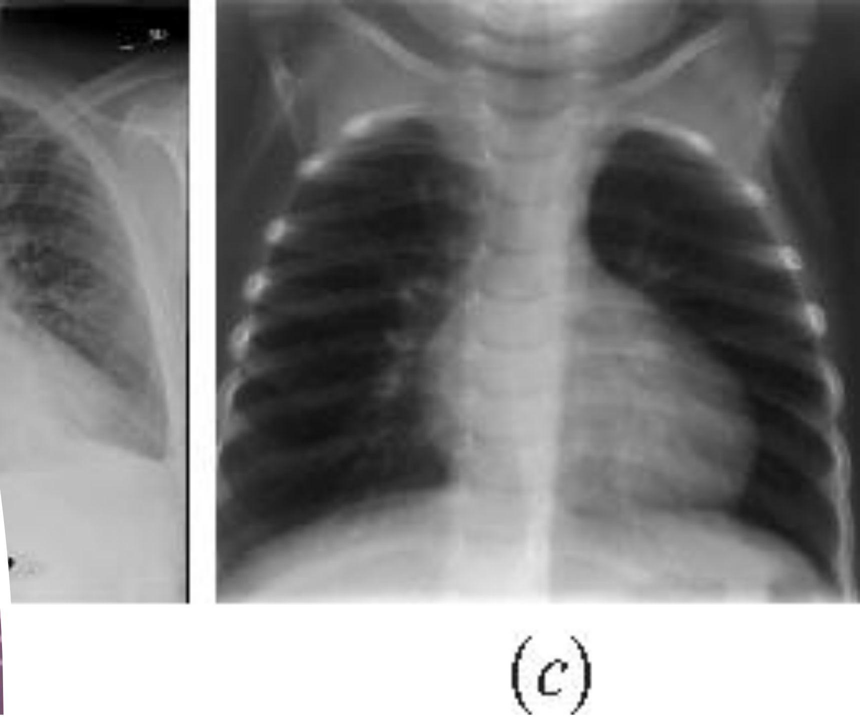
Springboard Data Science Career Track -
Capstone Two

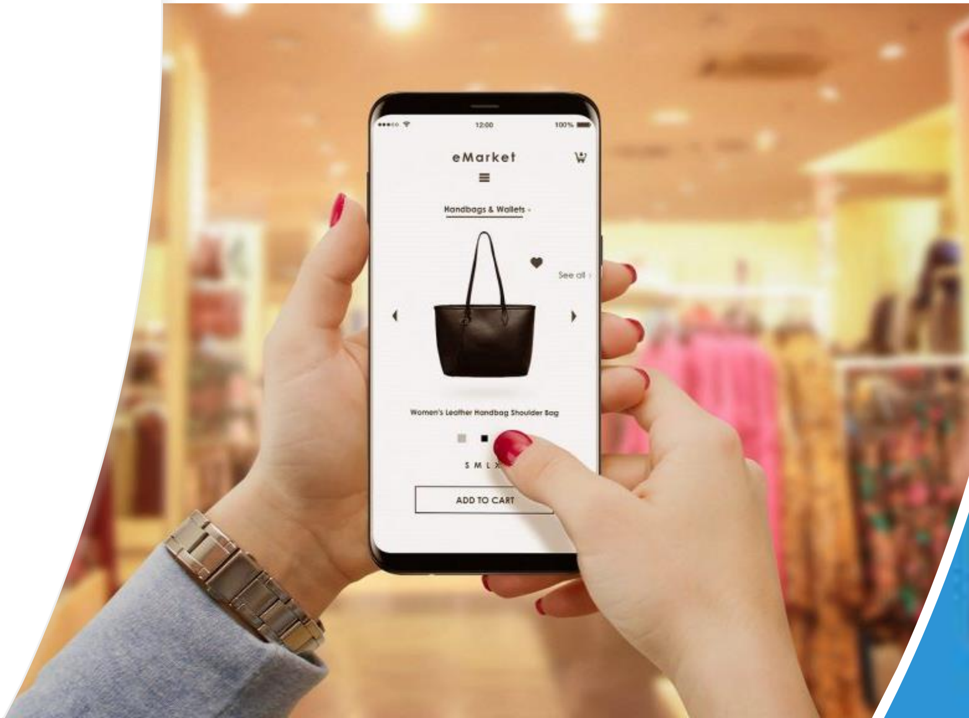
Rose Zdybel

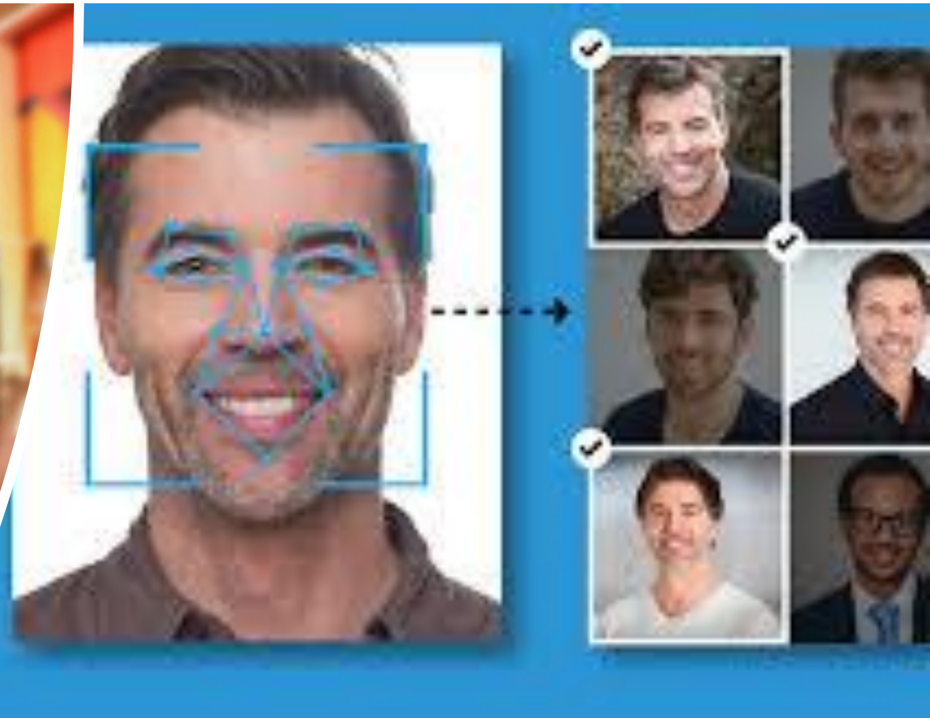
2020 July

Image Classification has a wide range of applications

- Medical applications
- Facial recognition
- Self driving cars image recognition
- Online retailer automatic image classification
- Product locater based on user pictures



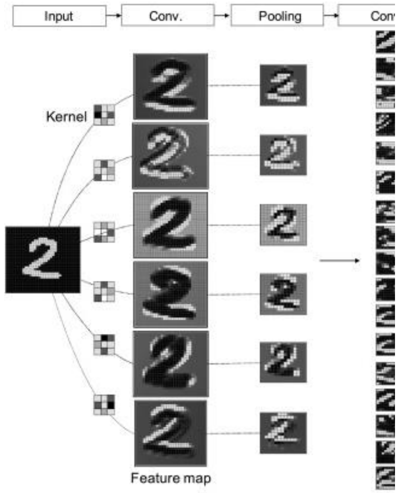


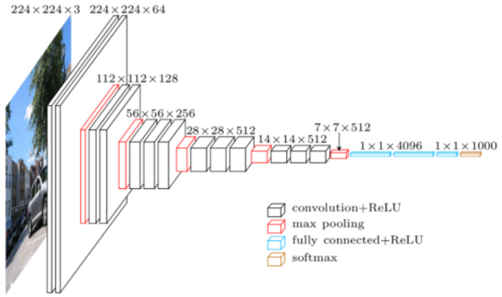


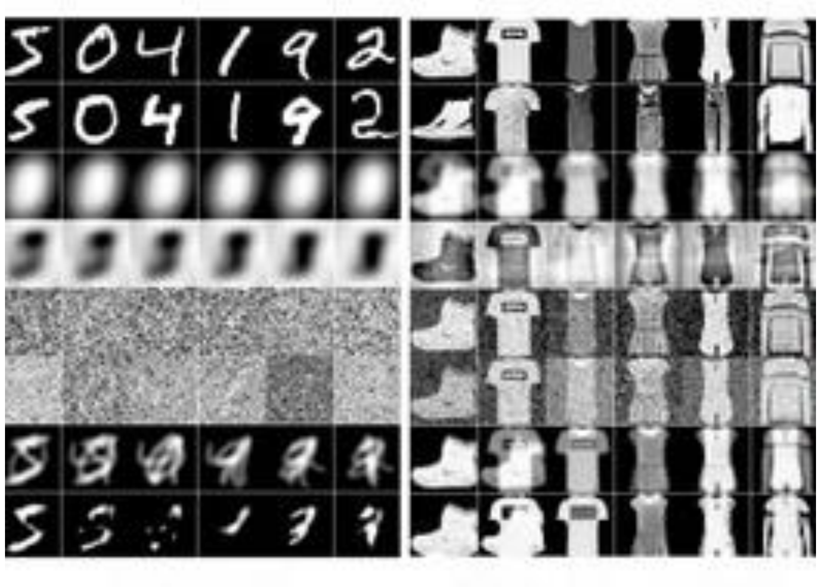
Project Objective: Image Classification Fundamentals

- Understand fundamental mechanisms used in image classification
  - To get experience in the challenges from training a model from scratch

- Maximize predictive capabilities







EDA

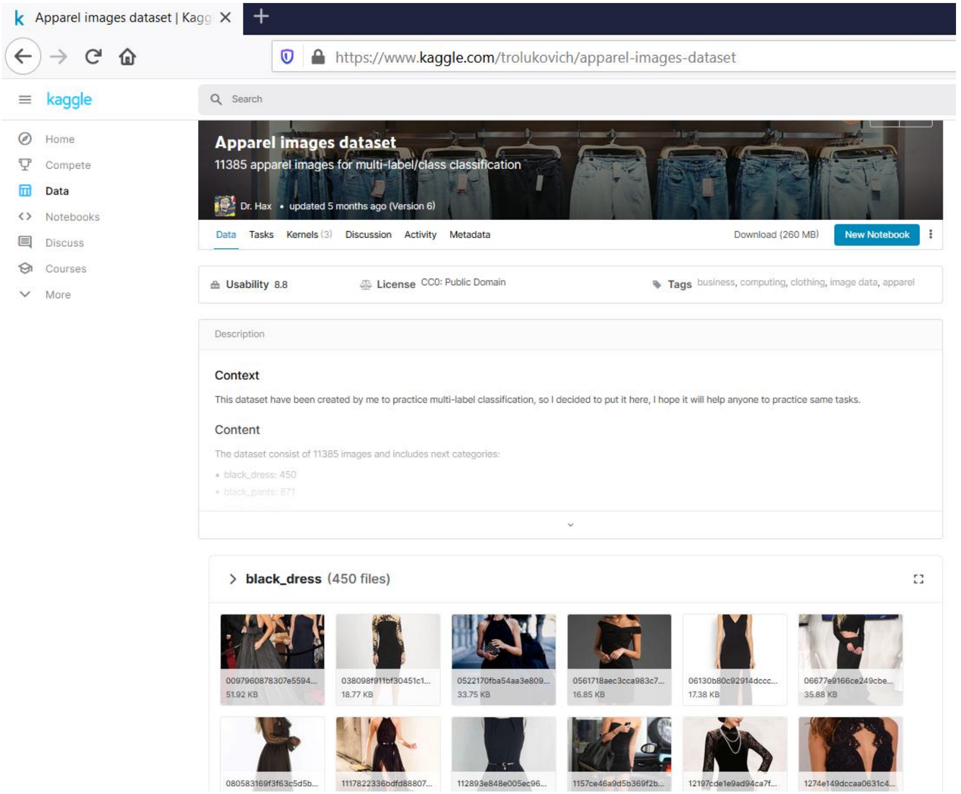
Exploratory Data Analysis

Image size

Categories

Data: Kaggle Apparel Image Data Set

- Subsetted due to resource constraints:
  - Compute:
    - Memory
    - GPU unavailability (initially)
  - Training time



Apparel images dataset

11385 apparel images for multi-label/class classification

Dr. Hax • updated 5 months ago (Version 6)

Usability 8.8

License CC0: Public Domain

Tags business, computing, clothing, image data, apparel

Description

Context

This dataset have been created by me to practice multi-label classification, so I decided to put it here, I hope it will help anyone to practice same tasks.

Content

The dataset consist of 11385 images and includes next categories:

- black_dress: 450
- black_pants: 871

black_dress (450 files)

0097960878307e5594... 51.92 KB	038098f911bf30451c1... 18.77 KB	0522170fba54aa3e809... 33.75 KB	0561718aec3cca983c7... 16.85 KB	06130b80c92914dccc... 17.38 KB	06677e9166ce249cbe... 35.88 KB
080583169f3f63c5d5b...	1117822336bdfd88807...	112893e848e005ec96...	1157ce46a9d5b369f2b...	12197cde1e9ad94ca7f...	1274e149dccaa0631c4...

We used categories with the largest number of samples

871	**black_pants**
818	**white_dress**
800	**red_dress**
798	**blue_pants**
766	**black_shoes**
741	**blue_shirt**
715	**black_shirt**
610	**red_shoes**
600	**white_shoes**
523	**blue_shoes**
502	blue_dress
464	brown_shoes
455	green_shoes
450	black_dress
328	black_shorts
311	brown_pants
308	red_pants
299	blue_shorts
274	white_pants
230	green_shirt
227	green_pants
135	green_shorts
120	white_shorts
40	brown_shorts

Black Shirt

White Dress

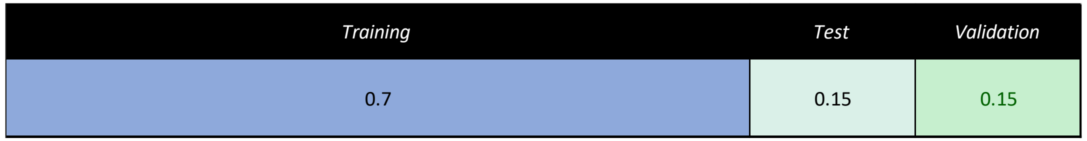
Red Dress

Sample data

Approach

“Experiments”

Data Splits



Training	*Test*	*Validation*
0.7	0.15	0.15

Training

70%

Train Model Parameters

Validation

15%

Used for Early Stopping

- Loss does not decrease within specified number of Epochs

Test

15%

Measure Model Performance

Initial exploratory results influenced approach

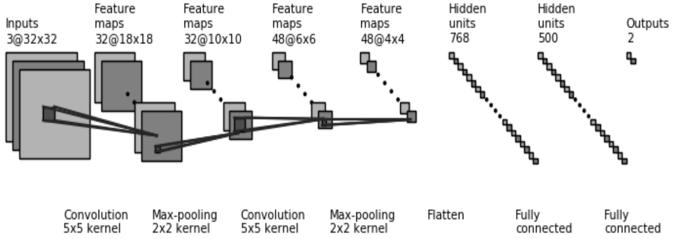
- Issues from initial model exploration on laptop:
    - Memory deficiency
    - Training time
        - Complex models
        - Large number of pixels
    - Lack of convergence

Change	Training time	Model Stability	CPU resources	Memory resources	Accuracy
Colab Env	+		+	+	
Data set size reduction	+				-
Image size reduction options	+			+	+/-
Optimizer/LR options		+			+/-

Approach: Investigate most influential parameters

Model Architecture

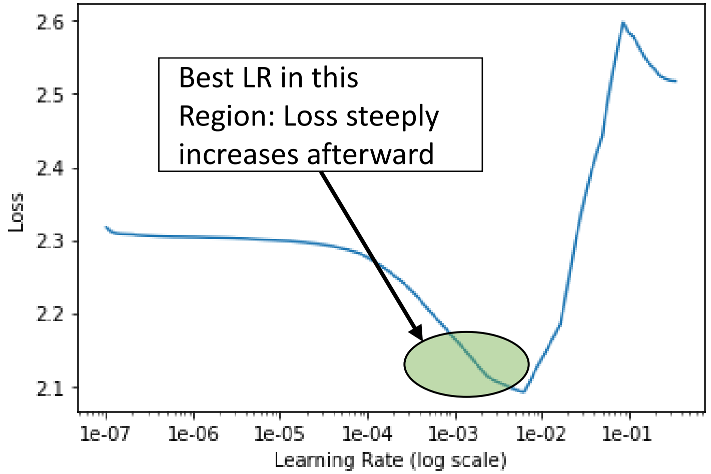
- Block* configuration
- # blocks
- # nodes per layer



****Block:** one or more convolution layers plus a pool layer*

Optimizers & Learning Rate

- Use LR-Finder
- Verified results on selected parameter combinations

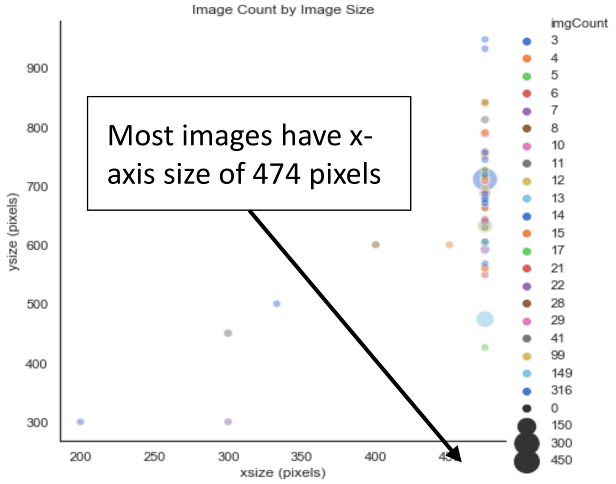


Best LR in this Region: Loss steeply increases afterward

***Example**: Model 3; AMSGRAD optimizer:*

Image Size

- Source size varies
- Use single size
- Larger size increases training time

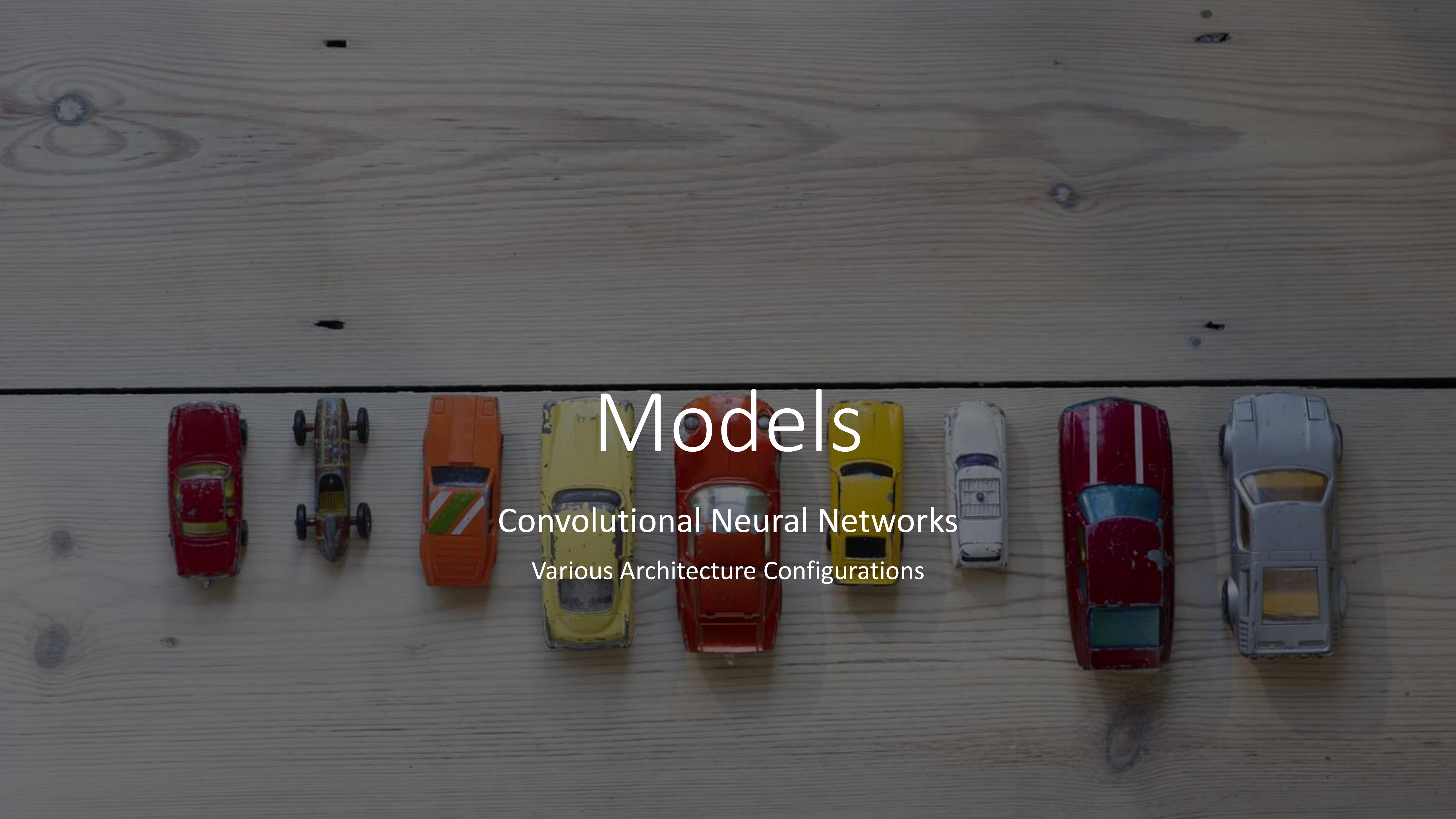


Most images have x-axis size of 474 pixels

Parameter Grid

- Machine Learning grid search-like mechanism
    - Ran experiments with a subset of parameters based on initial exploratory results
    - Shown later in results section

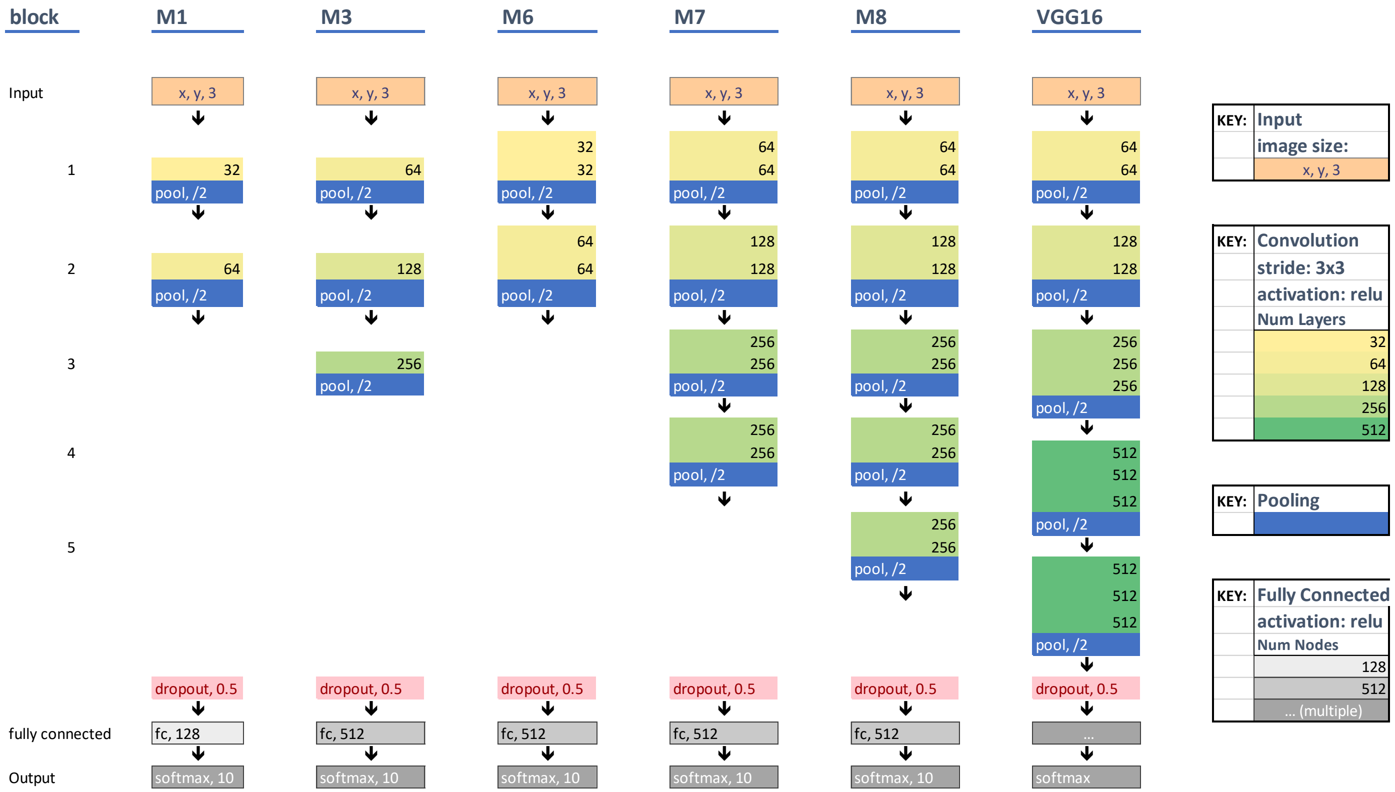
OPTION	VALUES
Optimizer	SGD , AMSGRAD(AMSGrad)
Learning Rate	.01, 0.001, .0001
Models	M1, M3, M6, M7, M8
Image Size (“pixels”)	150x150, 300x300



Models

Convolutional Neural Networks

Various Architecture Configurations



block	M1	M3	M6	M7	M8	VGG16
Input	x, y, 3	x, y, 3	x, y, 3	x, y, 3	x, y, 3	x, y, 3
1	32 pool, /2	64 pool, /2	32 32 pool, /2	64 64 pool, /2	64 64 pool, /2	64 64 pool, /2
2	64 pool, /2	128 pool, /2	64 64 pool, /2	128 128 pool, /2	128 128 pool, /2	128 128 pool, /2
3		256 pool, /2		256 256 pool, /2	256 256 pool, /2	256 256 256 pool, /2
4				256 256 pool, /2	256 256 pool, /2	512 512 512 pool, /2
5					256 256 pool, /2	512 512 512 pool, /2
	dropout, 0.5	dropout, 0.5	dropout, 0.5	dropout, 0.5	dropout, 0.5	dropout, 0.5
fully connected	fc, 128	fc, 512	fc, 512	fc, 512	fc, 512	...
Output	softmax, 10	softmax, 10	softmax, 10	softmax, 10	softmax, 10	softmax

KEY:	Input
	image size:
	x, y, 3

KEY:	Convolution
	stride: 3x3
	activation: relu
	Num Layers
	32
	64
	128
	256
	512

KEY:	Pooling

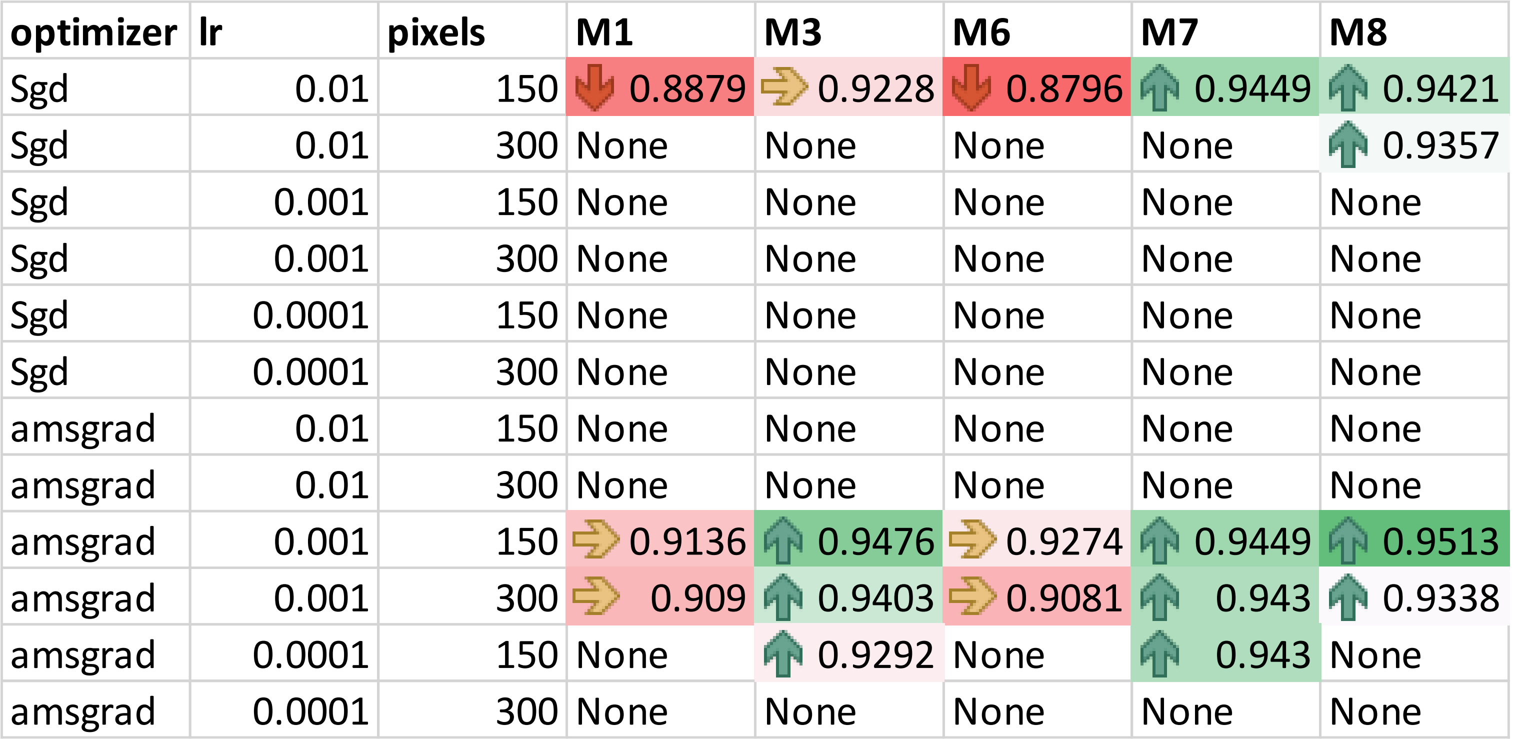
KEY:	Fully Connected
	activation: relu
	Num Nodes
	128
	512
	... (multiple)

Model Architectures

- VGG16 shown for reference
  - Did not train or use pretrained model
  - For future consideration

Results

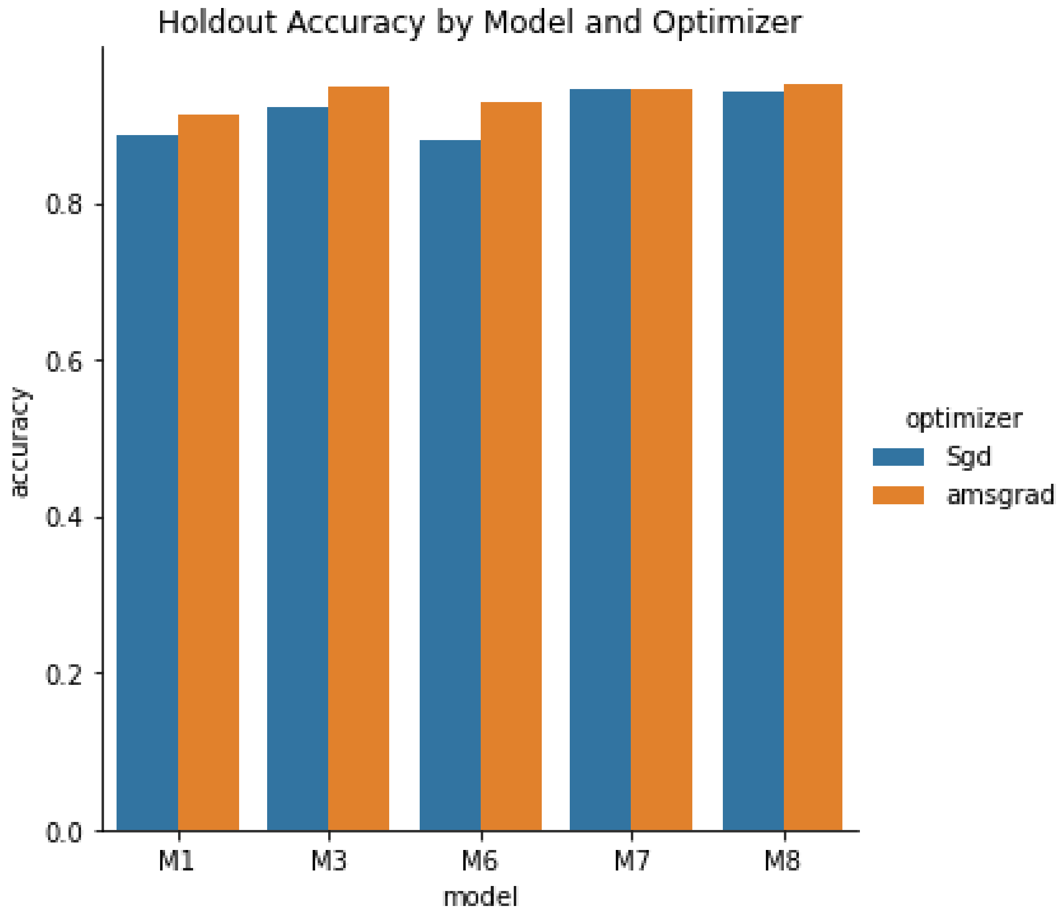
After Initial exploration, ran most promising parameter combinations



optimizer	lr	pixels	M1	M3	M6	M7	M8
Sgd	0.01	150	↓ 0.8879	→ 0.9228	↓ 0.8796	↑ 0.9449	↑ 0.9421
Sgd	0.01	300	None	None	None	None	↑ 0.9357
Sgd	0.001	150	None	None	None	None	None
Sgd	0.001	300	None	None	None	None	None
Sgd	0.0001	150	None	None	None	None	None
Sgd	0.0001	300	None	None	None	None	None
amsgrad	0.01	150	None	None	None	None	None
amsgrad	0.01	300	None	None	None	None	None
amsgrad	0.001	150	→ 0.9136	↑ 0.9476	→ 0.9274	↑ 0.9449	↑ 0.9513
amsgrad	0.001	300	→ 0.909	↑ 0.9403	→ 0.9081	↑ 0.943	↑ 0.9338
amsgrad	0.0001	150	None	↑ 0.9292	None	↑ 0.943	None
amsgrad	0.0001	300	None	None	None	None	None

Metric: Test (Holdout) Data Accuracy

Optimizer Comparison

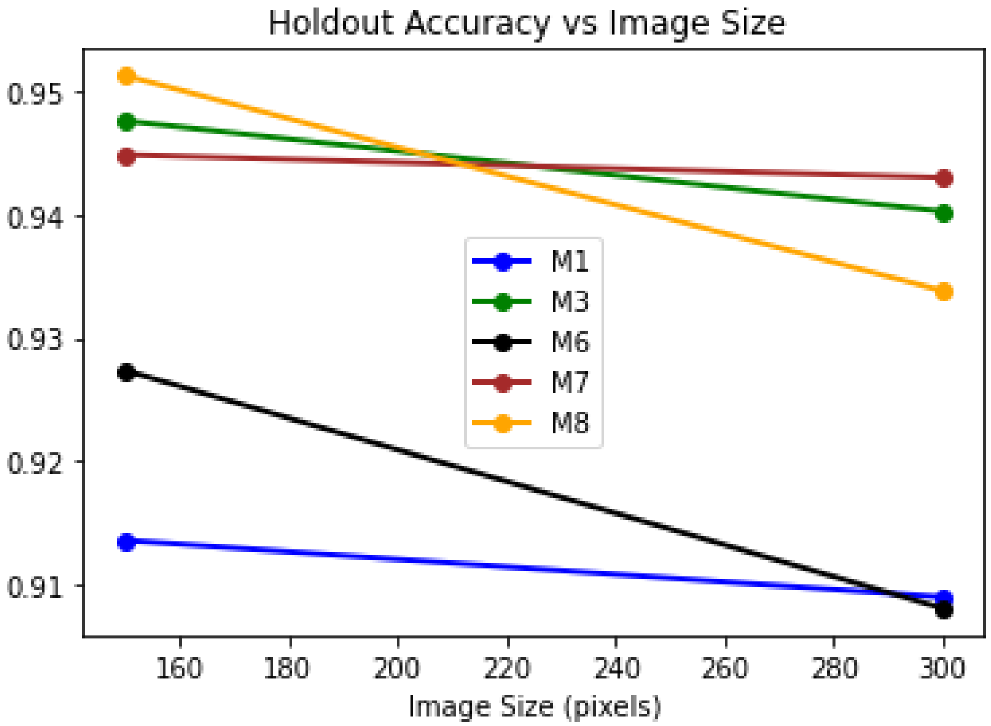


AMSGrad* did better or equivalent to SGD for all models
* **NOTE**: Adam optimizer with AMSGrad option

Image Size Comparison

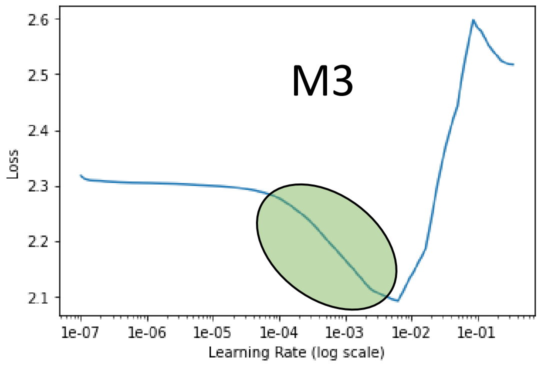
- We used two sizes
  - 150 x 150
  - 300 x 300

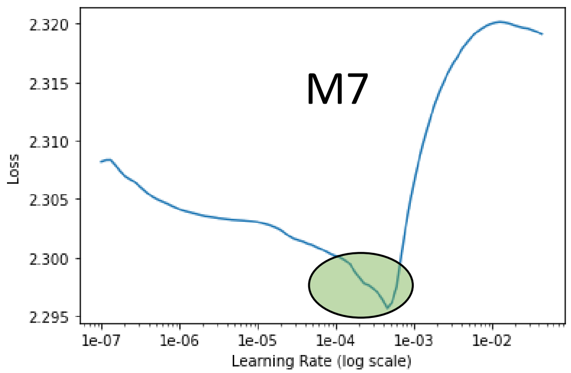
- Smaller size produced better results
  - Should consider smaller sizes in future work

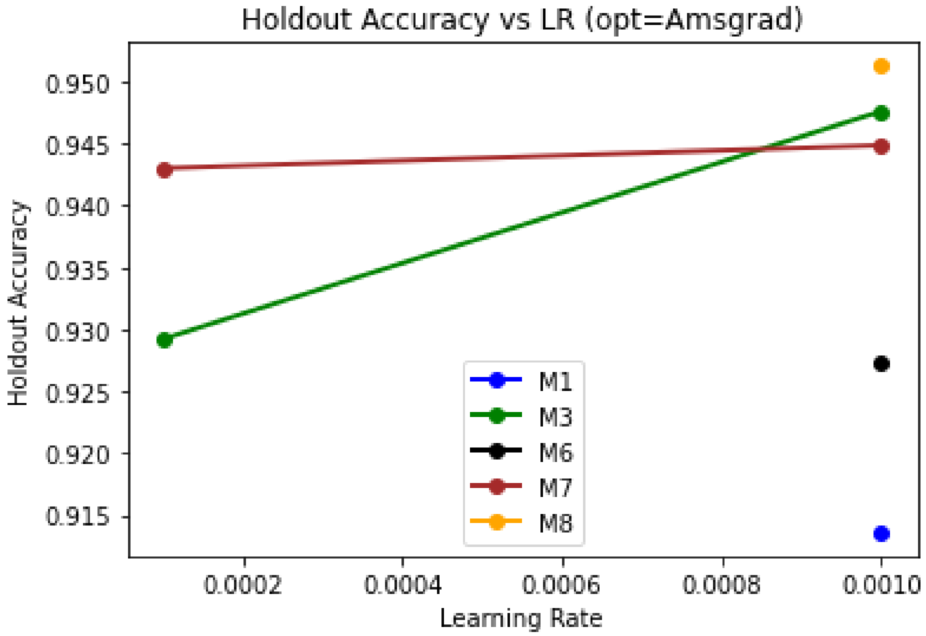


Learning Rate Comparison

- We selectively confirmed LR-finder recommendations:
  - Models
    - M3
    - M7
  - Optimizer: AMSGrad

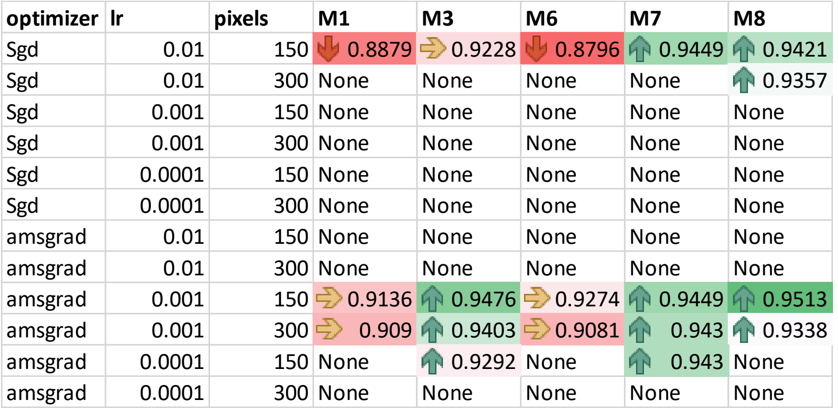






Best Model

Based on test data prediction accuracy

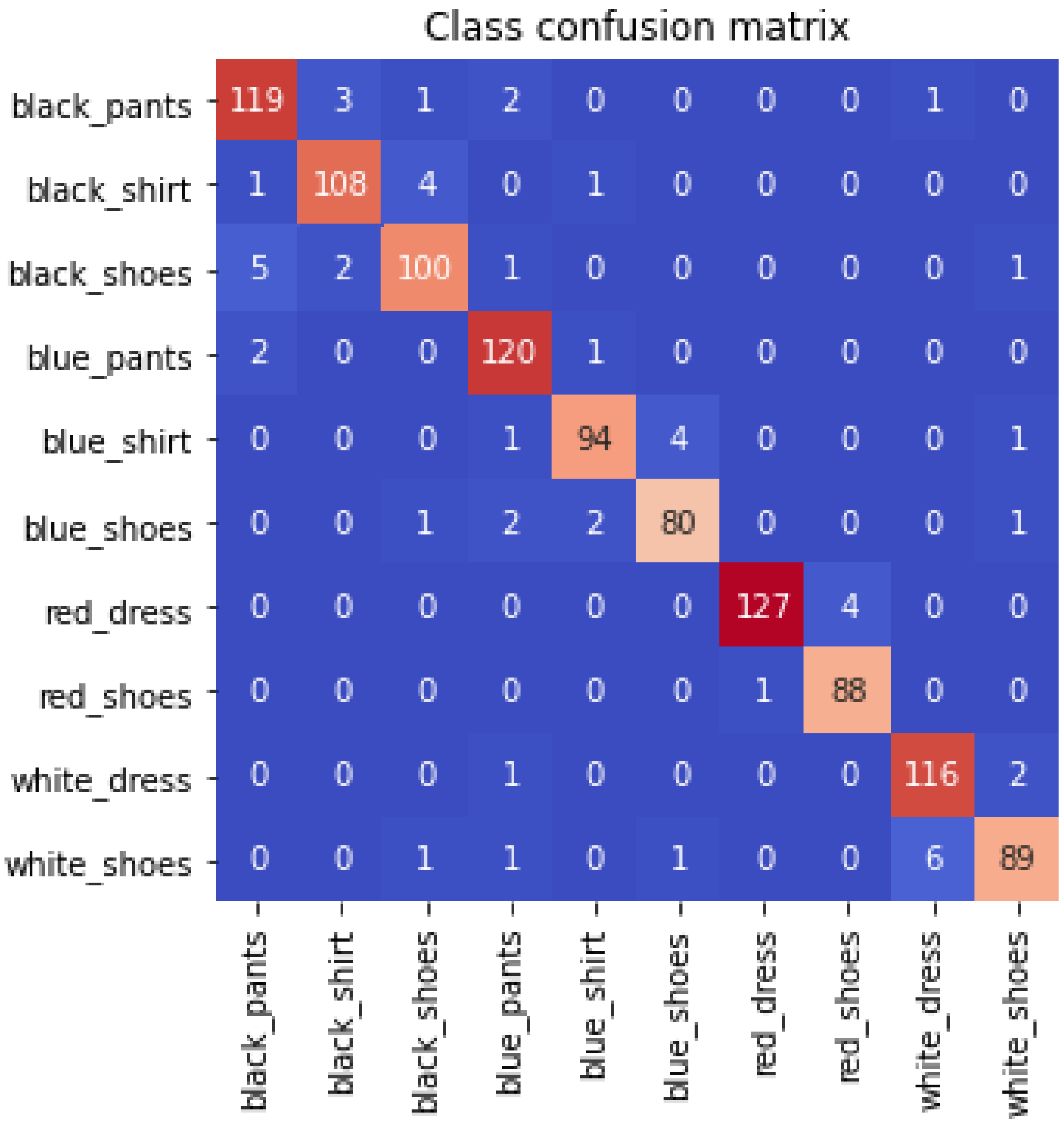


optimizer	lr	pixels	M1	M3	M6	M7	M8
Sgd	0.01	150	↓ 0.8879	⇨ 0.9228	↓ 0.8796	↑ 0.9449	↑ 0.9421
Sgd	0.01	300	None	None	None	None	↑ 0.9357
Sgd	0.001	150	None	None	None	None	None
Sgd	0.001	300	None	None	None	None	None
Sgd	0.0001	150	None	None	None	None	None
Sgd	0.0001	300	None	None	None	None	None
amsgrad	0.01	150	None	None	None	None	None
amsgrad	0.01	300	None	None	None	None	None
amsgrad	0.001	150	⇨ 0.9136	↑ 0.9476	⇨ 0.9274	↑ 0.9449	↑ 0.9513
amsgrad	0.001	300	⇨ 0.909	↑ 0.9403	⇨ 0.9081	↑ 0.943	↑ 0.9338
amsgrad	0.0001	150	None	↑ 0.9292	None	↑ 0.943	None
amsgrad	0.0001	300	None	None	None	None	None

Test Data Accuracy	0.951
Model	M8
Image Size	150 x 150
Optimizer	AMSGrad
Learning Rate	.001

Confusion Matrix

- Major Model issues
  - White shoes classified as white dress (6)
  - Black shoes classified as black pants (5)



Class confusion matrix

	black_pants	black_shirt	black_shoes	blue_pants	blue_shirt	blue_shoes	red_dress	red_shoes	white_dress	white_shoes
black_pants	119	3	1	2	0	0	0	0	1	0
black_shirt	1	108	4	0	1	0	0	0	0	0
black_shoes	5	2	100	1	0	0	0	0	0	1
blue_pants	2	0	0	120	1	0	0	0	0	0
blue_shirt	0	0	0	1	94	4	0	0	0	1
blue_shoes	0	0	1	2	2	80	0	0	0	1
red_dress	0	0	0	0	0	0	127	4	0	0
red_shoes	0	0	0	0	0	0	1	88	0	0
white_dress	0	0	0	1	0	0	0	0	116	2
white_shoes	0	0	1	1	0	1	0	0	6	89

Mis-classified image examples

- Actual | Predicted
  - White shoes | white dress
  - Black shoes | black pants

Conclusion

- Explored developing a Convolution Neural Network from scratch
- Explored the effect of influential parameters
- Used Colab due to local compute resource limitations
- Developed a model with 95% predictive accuracy
-
- Future work considerations
  - Investigate multi-class model
    - Potential for better predictive capability
  - Investigate more parameter values: image sizes, optimizers
  - Investigate models with more blocks/layers
  - Investigate use of one or more pre-trained models