The Effects of Dataset Conditions on The Predictive Accuracy of ResNet152 for Classifying *Yu-Gi-Oh! Trading Card Game* Cards & Artwork

Michael Mays
mfmays@wisc.edu

Daniel Halberg
dhhalberg@wisc.edu

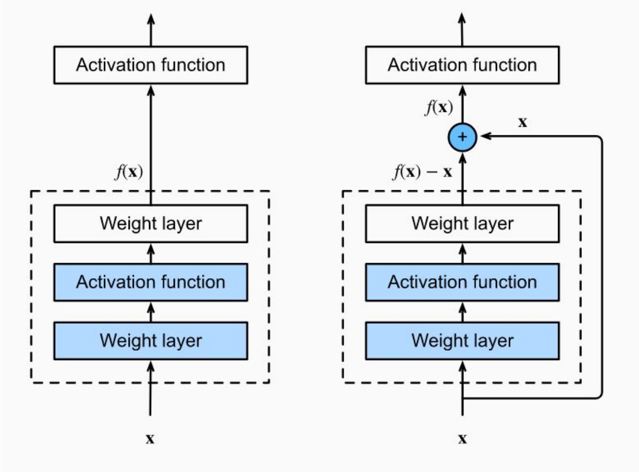
Department of Statistics
University of Wisconsin–Madison

April 29, 2021

Source:
https://storage.googleapis.com/ygoprodeck.com/pics/97631303.jpg

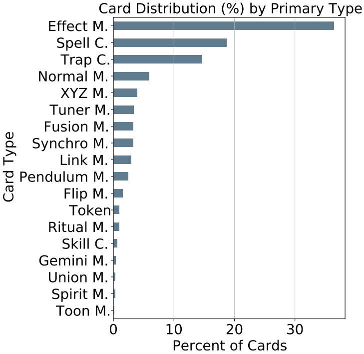
- Background & related work

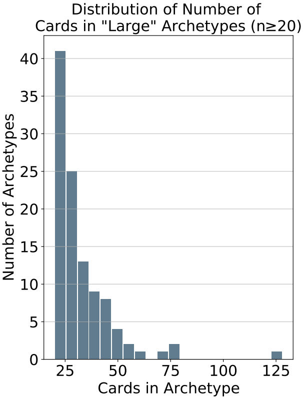
- Background & related work
- Proposed method: ResNet152



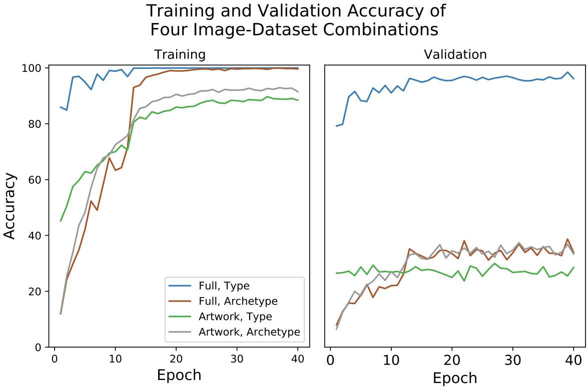
Overview

- Background & related work
- Proposed method: ResNet152
- Experiment: Design & dataset





- Background & related work
- Proposed method: ResNet152
- Experiment: Design & dataset
- Results



Background: The *Yu-Gi-Oh! Trading Card Game*

Figure: Source:
`https://storage.googleapis.com/ygoprodeck.com/pics/10000.jpg`

Background: The *Yu-Gi-Oh! Trading Card Game*

- Launched 2002 in North America

Figure: Source:
https://storage.googleapis.com/ygoprodeck.com/pics/10000.jpg

Background: The *Yu-Gi-Oh! Trading Card Game*

- Launched 2002 in North America
- Over 10,000 distinct cards

Figure: Source: `https://storage.googleapis.com/ygoprodeck.com/pics/10000.jpg`

Background: The *Yu-Gi-Oh! Trading Card Game*

- Launched 2002 in North America
- Over 10,000 distinct cards
- Numerous ways to classify each card. We use:

Figure: Source: `https://storage.googleapis.com/ygoprodeck.com/pics/10000.jpg`

Background: The *Yu-Gi-Oh! Trading Card Game*

- Launched 2002 in North America
- Over 10,000 distinct cards
- Numerous ways to classify each card. We use:
  - **Primary type**: Determines how the card functions

Figure: Source:
`https://storage.googleapis.com/ygoprodeck.com/pics/10000.jpg`

Background: The *Yu-Gi-Oh! Trading Card Game*

- Launched 2002 in North America
- Over 10,000 distinct cards
- Numerous ways to classify each card. We use:
    - **Primary type**: Determines how the card functions
    - **Archetype**: Which "family" the card belongs to (if any)

Figure: Source: `https://storage.googleapis.com/ygoprodeck.com/pics/10000.jpg`

- Ability of a pre-trained ResNet152 CNN to classify *YGO* cards using datasets with varied:

- Ability of a pre-trained ResNet152 CNN to classify *YGO* cards using datasets with varied:
  - Size & classification target

- Ability of a pre-trained ResNet152 CNN to classify *YGO* cards using datasets with varied:
    - Size & classification target
    - Information density

The Goal

- Ability of a pre-trained ResNet152 CNN to classify *YGO* cards using datasets with varied:
  - Size & classification target
  - Information density
- Why?

- Ability of a pre-trained ResNet152 CNN to classify *YGO* cards using datasets with varied:
  - Size & classification target
  - Information density
- Why?
  - Why not?

The Goal

- Ability of a pre-trained ResNet152 CNN to classify *YGO* cards using datasets with varied:
  - Size & classification target
  - Information density
- Why?
  - Why not?
  - To demonstrate the effect of various dataset conditions on ResNet152's ability to accurately classify YGO cards.

The Goal

- Ability of a pre-trained ResNet152 CNN to classify *YGO* cards using datasets with varied:
  - Size & classification target
  - Information density
- Why?
  - Why not?
  - To demonstrate the effect of various dataset conditions on ResNet152's ability to accurately classify YGO cards.
  - Tons of literature about dataset effects *in general*, but this (*YGO*) is a novel dataset.

- Tons of comparisons of different network architectures' accuracies on a given data set (e.g., [1]; [2]; [3]).

Related Work: CNNs

- Tons of comparisons of different network architectures' accuracies on a given data set (e.g., [1]; [2]; [3]).
- Same for dataset conditions' impact on a given network's ability to accuracy classify ([4]; [5]; [6]; and [7], to name a few).

- Tons of comparisons of different network architectures' accuracies on a given data set (e.g., [1]; [2]; [3]).
- Same for dataset conditions' impact on a given network's ability to accuracy classify ([4]; [5]; [6]; and [7], to name a few).
- Fundamental, so we will assume general knowledge of these results:

- Tons of comparisons of different network architectures' accuracies on a given data set (e.g., [1]; [2]; [3]).
- Same for dataset conditions' impact on a given network's ability to accuracy classify ([4]; [5]; [6]; and [7], to name a few).
- Fundamental, so we will assume general knowledge of these results:
  - More data tends to improve predictive accuracy (up to a point).

Related Work: CNNs

- Tons of comparisons of different network architectures' accuracies on a given data set (e.g., [1]; [2]; [3]).
- Same for dataset conditions' impact on a given network's ability to accuracy classify ([4]; [5]; [6]; and [7], to name a few).
- Fundamental, so we will assume general knowledge of these results:
  - More data tends to improve predictive accuracy (up to a point).
  - Newer network architectures tend to perform better than older ones (up to a point).

Related Work: CNNs

- Tons of comparisons of different network architectures' accuracies on a given data set (e.g., [1]; [2]; [3]).
- Same for dataset conditions' impact on a given network's ability to accuracy classify ([4]; [5]; [6]; and [7], to name a few).
- Fundamental, so we will assume general knowledge of these results:
  - More data tends to improve predictive accuracy (up to a point).
  - Newer network architectures tend to perform better than older ones (up to a point).
  - And so on.

- The *Yu-Gi-Oh! Neuron* phone application includes augmented reality card recognition [8].
- Lowhur sought to imitate its functionality via deep neural network one-shot learning [9].
- GitHub user `chronoreaper` used deep learning to train an AI to play a *YGO* video game, including some card recognition [10].

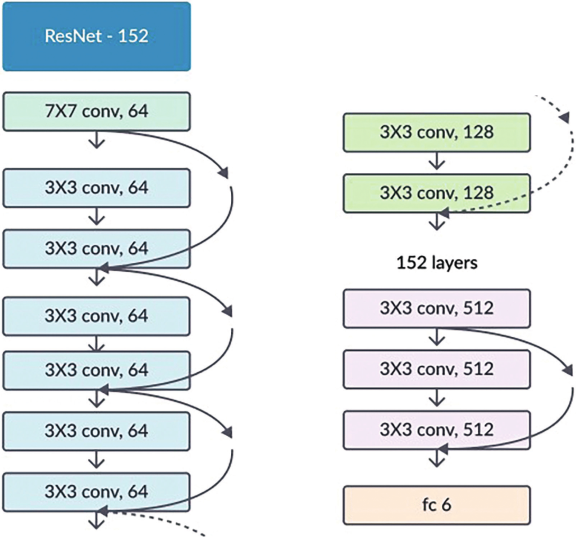


Figure: ResNet152 architecture diagram. Source: [11], split for space reasons.

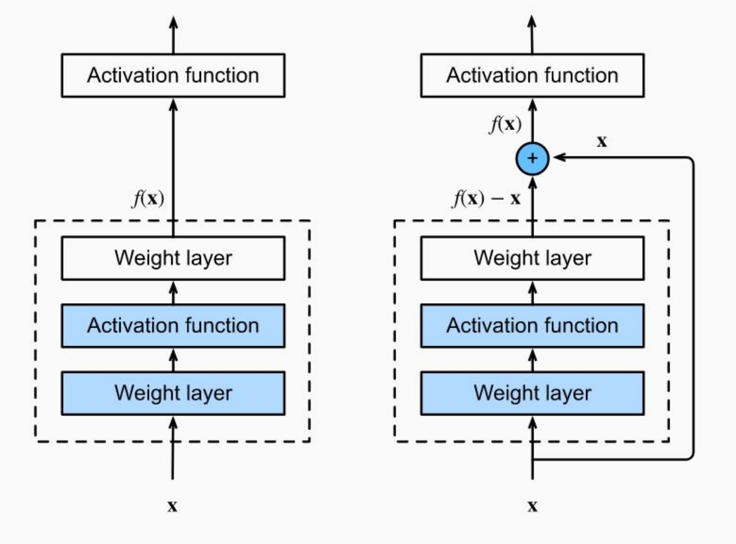


Figure: Diagram of residual block architecture. Source: [12]

ResNet152 Details

Hyperparameter/ Setting	Value(s)	Comment
Random seed	453	
Batch size	32	
Epochs	40	
Optimizer	Adam	
Learning rate	Full cards, primary type: 0.0005 Full cards, archetype: 0.0003 Artwork, primary type: 0.0001 Artwork, archetype: 0.0001	
Scheduler	Reduce LR on plateau Factor: 0.2	New $=$ Old $\times$ Factor

Figure: Model hyperparameter values.

Hyperparameter/ Setting	Value(s)	Comment
Random seed	453	
Batch size	32	
Epochs	40	
Optimizer	Adam	
Learning rate	Full cards, primary type: 0.0005 Full cards, archetype: 0.0003 Artwork, primary type: 0.0001 Artwork, archetype: 0.0001	
Scheduler	Reduce LR on plateau Factor: 0.2	New = Old $\times$ Factor

Figure: Model hyperparameter values.

Note: Learning rate was tuned separately for each experiment to give each network a 'fair' chance.

- **Dataset size & classification task**: For a given image type (either full cards or artwork only), how do the size of the *YGO* dataset (all cards vs. 'large' archetype cards) and classification task (either primary type or 'large' archetypes) jointly impact ResNet152's classification accuracy?

- **Dataset size & classification task**: For a given image type (either full cards or artwork only), how do the size of the *YGO* dataset (all cards vs. 'large' archetype cards) and classification task (either primary type or 'large' archetypes) jointly impact ResNet152's classification accuracy?
- **Image type**: For a given size of *YGO* dataset (either all cards or 'large' archetype cards) and classification task (either primary type or 'large' archetypes), how does image type (full cards vs. artwork only) impact ResNet152's classification accuracy?

		Target (Data subset)	
Variable	Image	Primary Type (All cards)	Archetype ('Large' arch.)
No. classes		Less	More
Dataset size		More	Less
Image info.	Full card	More	
	Artwork	Less	

Figure: Experimental parameters. Rows represent the two different information densities (full vs. artwork). Columns represent the two different dataset sizes (all vs. 'large' archetypes).

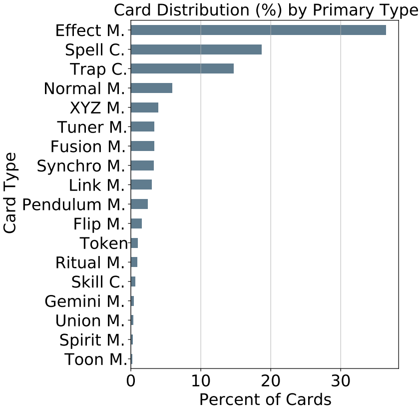
Image	Variable	Target (Data subset): Primary Type (All cards)	Target (Data subset): Archetype ('Large' arch.)
	No. classes	18 classes	107 classes
	Dataset size	11,149 images	3,451 images
Full card	Image size	$614H \times 422W$	
Artwork	Image size	Normal: $320H \times 322W$ Pendulum: $272H \times 367W$	

Figure: Experimental parameters' values.

Dataset: Data augmentation

Transform	Value(s)/Range	Comment
Resize	Full card: 312x211	HxW in px, half-sized
Random resized crop	Artwork: 272x322	HxW in px, each is the smaller image format's value for that dimension. This is akin to randomly shifting pendulum cards horizontally and non-pendulum cards vertically before cropping.
Random color jitter	Brightness: (0.75, 1.5) Contrast: (0.75, 1.5) Saturation: (0.75, 1.5) Hue: (0.9, 1.1)	Multiplier uniformly chosen from range (min, max)
Random flip	Horizontal: $p = 0.5$ Vertical: $p = 0.5$	
Random affine transformation	Interpolation: Bilinear	
	Translate: (0.2, 0.2)	Max. (H, W) shift (prop. of size)
	Shear: (0, 10)	Degrees, both x and y (separately)
Normalize	Mean: (0.485, 0.456, 0.406) Std. dev.: (0.229, 0.224, 0.225)	(R, G, B), see `https://pytorch.org/vision/stable/models.html`

Figure: Data augmentation settings.



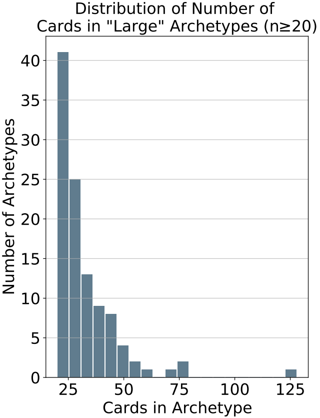


Figure: Left: Distribution of primary types. "M." is short for "Monster" and "C." is short for "Card".; Right: Distribution of archetype size among 'large' archetypes ($n \geq 20$).

Hardware & Software

Hardware & Software

- Google Colab

Hardware & Software

- Google Colab
- Used the pre-trained ResNet152 network in `torchvision`.

Hardware & Software

- Google Colab
- Used the pre-trained ResNet152 network in `torchvision`.
- Since both datasets are heavily class-imbalanced, all data loaders included the `ImbalancedDatasetSampler` found in `torchsampler` [13].

- Google Colab
- Used the pre-trained ResNet152 network in `torchvision`.
- Since both datasets are heavily class-imbalanced, all data loaders included the `ImbalancedDatasetSampler` found in `torchsampler` [13].
- Data wrangling was done with `Pandas` and `NumPy`.
- Data splitting was done with `train_test_split` in `sklearn`.
- Card data objects were serialized using `pickle`.
- Plots were created using `matplotlib` based on modified code from Dr. Sebastian Raschka's `helper_plotting.py` [14].
- 'Wrapper' functions and classes were created that call functions from Dr. Sebastian Raschka in `helper_dataset.py`, `helper_evaluation.py`, and `helper_train.py` [14].

Results

- We will go network-by-network.
- For each, we will discuss loss & accuracy curves.
- We will discuss as we go.
- Then summarize.

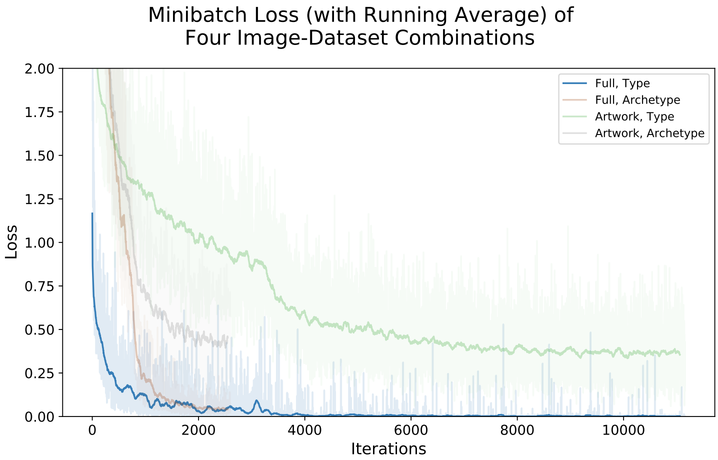


Figure: Minibatch loss curves with running average for ResNet152 under each set of image–dataset conditions.

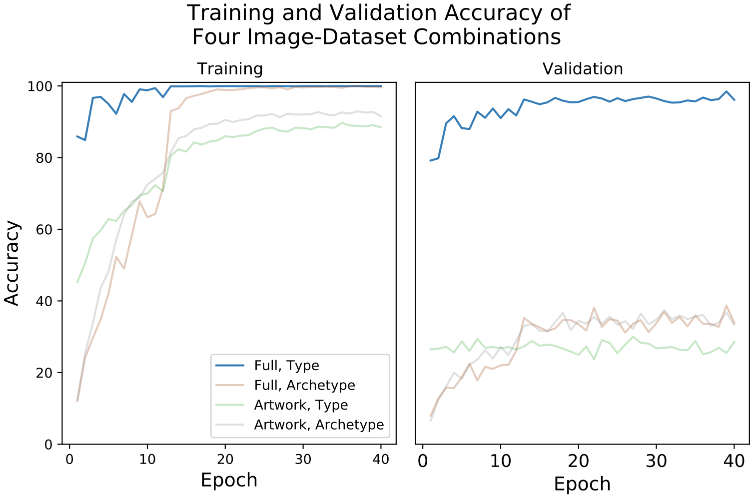


Figure: Training (left) and validation (right) accuracy curves for ResNet152 under each set of image–dataset conditions.

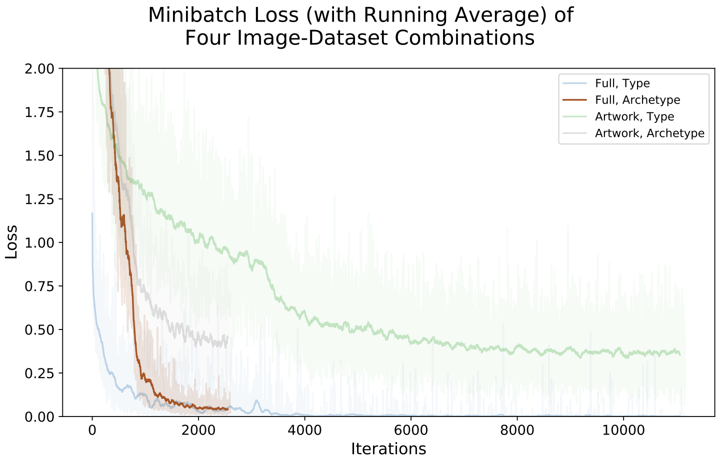


Figure: Minibatch loss curves with running average for ResNet152 under each set of image–dataset conditions.

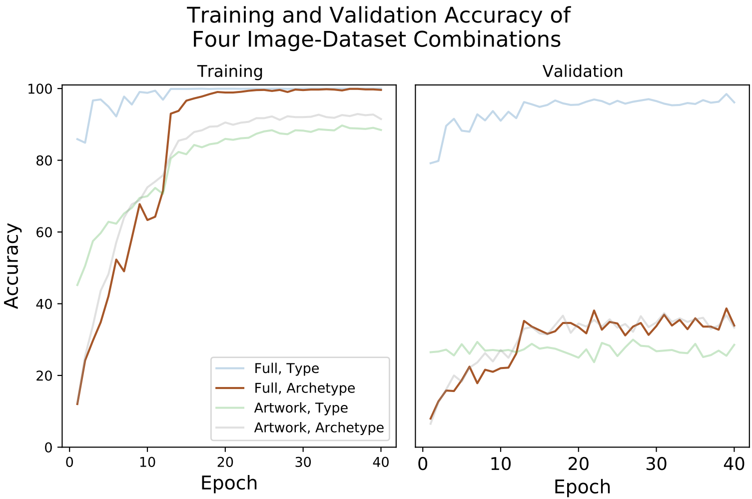


Figure: Training (left) and validation (right) accuracy curves for ResNet152 under each set of image–dataset conditions.

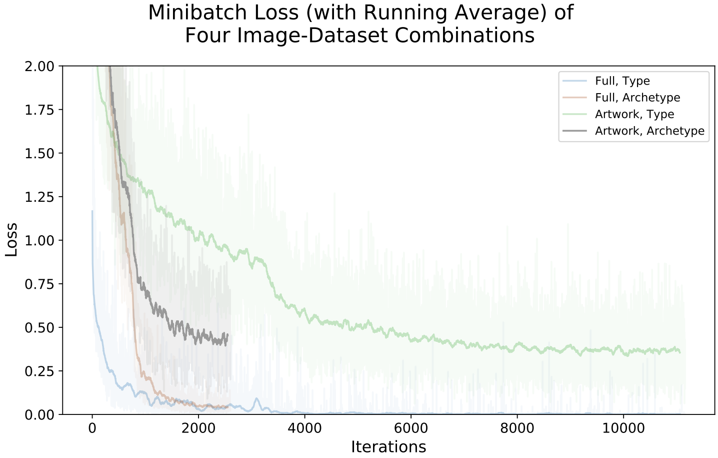


Figure: Minibatch loss curves with running average for ResNet152 under each set of image–dataset conditions.

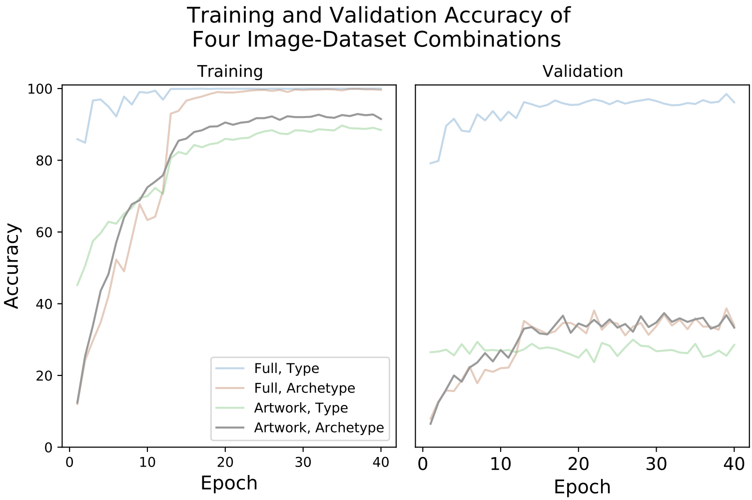


Figure: Training (left) and validation (right) accuracy curves for ResNet152 under each set of image–dataset conditions.

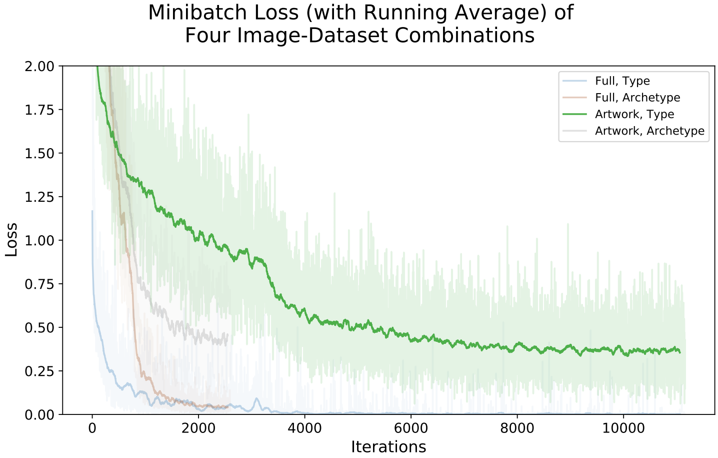


Figure: Minibatch loss curves with running average for ResNet152 under each set of image–dataset conditions.

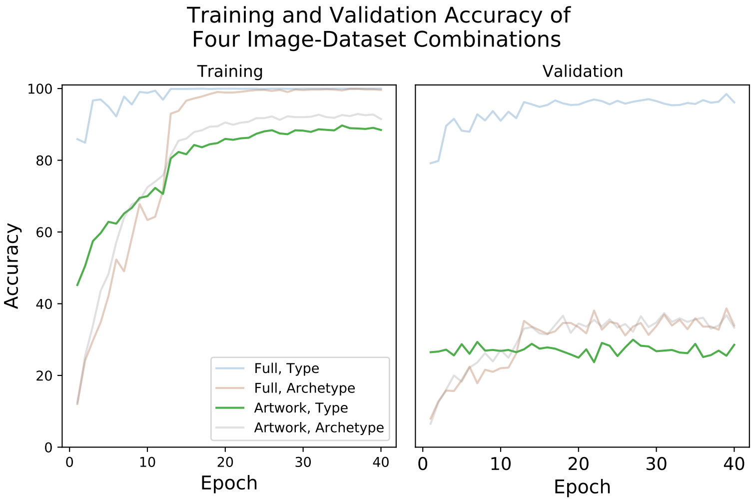


Figure: Training (left) and validation (right) accuracy curves for ResNet152 under each set of image–dataset conditions.

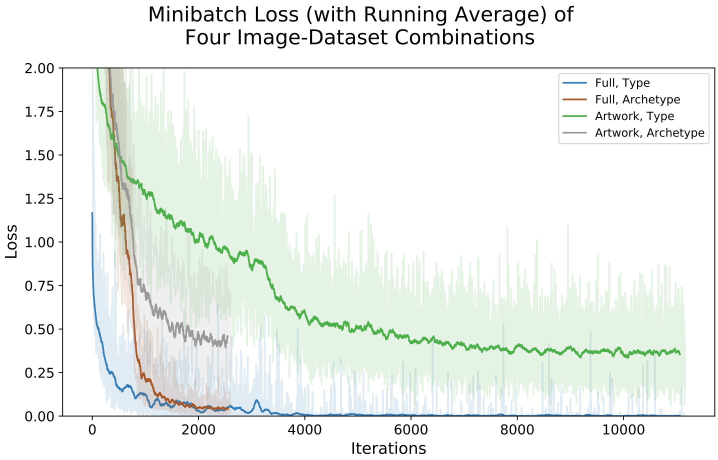


Figure: Minibatch loss curves with running average for ResNet152 under each set of image–dataset conditions. Note that the gray and brown curves (Artwork, Archetype and Full, Archetype, respectively) end ‘prematurely’ because these networks were trained on a smaller dataset for the same number of epochs as all other networks. An iteration is one epoch–batch, so for a fixed number of epochs, smaller datasets are trained for fewer total iterations.

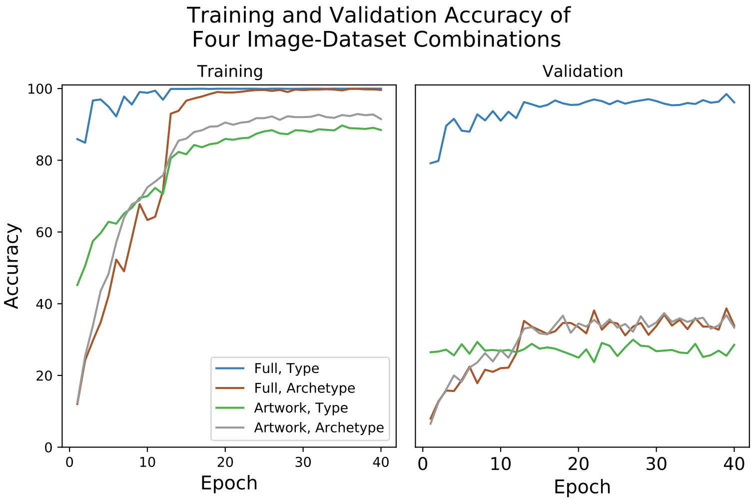


Figure: Training (left) and validation (right) accuracy curves for ResNet152 under each set of image–dataset conditions.

- **Image type**: One result is trivial and one is inconclusive.

Conclusion

- **Image type**: One result is trivial and one is inconclusive.
    - Obviously, the full cards were better than the artwork at Primary Type classification. This is a baseline.

Conclusion

- **Image type**: One result is trivial and one is inconclusive.
  - Obviously, the full cards were better than the artwork at Primary Type classification. This is a baseline.
  - Two archetype networks had near-identical validation accuracy despite having different image types $\implies$ dataset size and classification task (jointly) were the bottleneck.

- **Image type**: One result is trivial and one is inconclusive.
    - Obviously, the full cards were better than the artwork at Primary Type classification. This is a baseline.
    - Two archetype networks had near-identical validation accuracy despite having different image types $\implies$ dataset size and classification task (jointly) were the bottleneck.
    - But even a big dataset isn't enough. Even with all cards' artwork, ResNet couldn't classify Primary Type—there was no *relevant* information.

- **Image type**: One result is trivial and one is inconclusive.
  - Obviously, the full cards were better than the artwork at Primary Type classification. This is a baseline.
  - Two archetype networks had near-identical validation accuracy despite having different image types $\implies$ dataset size and classification task (jointly) were the bottleneck.
  - But even a big dataset isn't enough. Even with all cards' artwork, ResNet couldn't classify Primary Type—there was no *relevant* information.
- **Dataset size & classification task**: Jointly, these were the primary determinant of ResNet152's ability to classify *YGO* images.
  - There weren't enough images in the dataset to train ResNet152 for a 107-class classification task, regardless of image.
  - The validation accuracies represent a 'boundary condition' for ResNet152 on this dataset–target combination.

[1] K Bressem, L Adams, C Erxleben, B Hamm, S Niehues, and J Vahldiek. Comparing different deep learning architectures for classification of chest radiographs. *Scientific Reports*, (1):10, 2020. doi: 10.1038/s41598-020-70479-z.

[2] Yue Zhang. Evaluation of cnn models with fashion mnist data, 2019.

[3] Abhishek Mukhopadhyay, Pradipta Biswas, Ayush Agarwal, and Imon Mukherjee. Performance comparison of different cnn models for indian road dataset. In *Proceedings of the 2019 3rd International Conference on Graphics and Signal Processing*, ICGSP '19, page 2933, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450371469. doi: 10.1145/3338472.3338480. URL https://doi.org/10.1145/3338472.3338480.

[4] Chao Luo, Xiaojie Li, Lutao Wang, Jia He, Denggao Li, and Jiliu Zhou. How does the data set affect cnn-based image classification performance? In *2018 5th International Conference on Systems and Informatics (ICSAI)*, pages 361–366, 2018. doi: 10.1109/ICSAI.2018.8599448.

[5] Prasun Roy, Subhankar Ghosh, Saumik Bhattacharya, and Umapada Pal. Effects of degradations on deep neural network architectures, 2019.

[6] Huu Nhan Nguyen and Chanho Lee. Effects of hyper-parameters and dataset on cnn training. *Journal of The Institute of Korean Electrical and Electronics Engineers*, 22.1:14–20, 2018.

[7] Wei Zhao. Research on the deep learning of the small sample data based on transfer learning. *AIP Conference Proceedings*, 1864(1):020018, 2017. doi: 10.1063/1.4992835. URL `https://aip.scitation.org/doi/abs/10.1063/1.4992835`.

[8] Yu-gi-oh! neuron available worldwide from today for ios and android, Jul 2020. URL `https://www.konami.com/games/eu/en/topics/15555/`.

[9] Anthony Lowhur. I made an ai to recognize over 10,000 yugioh cards, Dec 2020. URL `https://towardsdatascience.com/i-made-an-ai-to-recognize-over-10-000-yugioh-cards-26fc6aed1588`.

[10] chronoreaper. YugiohAi, 2020. https://github.com/chronoreaper/YugiohAi.

[11] Stjepan Lonjak, Tin Kramberger, Ivan Cesar, and Renata Kovaevi. Automobile classification using transfer learning on resnet neural network architecture. 01 2020. doi: 10.19279/TVZ.PD.2020-8-1-18.

[12] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2020. `https://d2l.ai`.

[13] Ming Yang. Imbalanced dataset sampler, 2021. https://github.com/ufoym/imbalanced-dataset-sampler.

[14] Sebastian Raschka. L13, March 2021. https://github.com/rasbt/stat453-deep-learning-ss21/tree/main/L13/code.