Concatenated MNIST (CMNIST)

Making 784 pixels challenging again

Simon Wenkel*

September 24, 2019

The MNIST dataset is around for 25 years, it remains a standard benchmark in many publications. Modern approaches reach error rates of <1% on this dataset, however small variations can lead to significant accuracy losses. To motivate the research community to research and develop better algorithms for representation learning, we introduce CMNIST. CMNIST is a dataset that contains 134 subsets which are generated from existing MNIST-like datasets by concatenating them to provide a much more challenging dataset while keeping it within 784 pixels (28x28). Furthermore, we provide baseline results for some of the CMNIST subsets.

1 Introduction

Using deep neural networks is the state-of-the-art approach to classify images. The MNIST dataset [1] is 25 years old [2] and was a reasonable challenging problem back in the days. It still remains a reasonable challenging dataset for many classical machine learning algorithms [3]. However, with the rise of deep neural networks and almost ubiquitous application of deep convolutional neural networks it became the "hello world" of machine learning. Nevertheless, it remains one of the most popular benchmark datasets used in many state-of-the-art machine learning publications [4].

A general question about machine learning datasets is the question if a dataset is obsolete or, in other words, can we define a point at which a dataset is no longer useful as a benchmark because the underlying problem could be considered as solved. With MNIST, we could agree on calling it obsolete because error rates are really low ($< 1\%$). From a practical, non-research perspective these error rates are low enough to question the meaningfulness of further improvements. While developing and testing new algorithms and model architectures, we have to keep in mind that we are aiming to use representation learning and not dataset memorization. However, when applying small variations to the MNIST dataset (e.g. adding noise, affine transformations, etc.) or when testing a model (trained with MNIST images) on a larger test set, we observe a significant drop in model accuracies [2, 5–7]. We could interpret this as overfitting to the MNIST dataset similar to what have been shown for CIFAR-10 and ImageNet [8, 9]. This assumption is proved by Yadav & Bottou [2] as a result of reconstructing the original test set MNIST is based on, which itself resulted in the QMNIST dataset (Section 3.2). Therefore, we have to question if the (current) research on computer vision and usage of MNIST really leads to (better) representation learning or, accidentally, results in memorization.

There are ongoing discussions about if shapes or textures are dominant to a deep convolutional neural network. While many researchers came to the conclusion that shape is more dominant, Geirhos et al. [10] showed that texture might be much more dominant. This sensitivity to

*CryoPred OÜ, Tallinn, Estonia
simon{}simonwenkel.com

textures would explain adversarial attacks much better because many look invisible for humans. Considering that many neural networks are pre-trained on a subset of 1000 classes of ImageNet [11], we can assume that many of their limitations and failure cases are similar. Furthermore, research shows that some models performing extremely well on ImageNet do not necessarily outperform other approaches on MNIST. Nevertheless, many pre-trained models score high on MNIST-like datasets (Figure 1.1).

In contrast, a rather simple experiment of concatenating three MNIST-like datasets[1], namely EMNIST-MNIST, EMNIST-Letters (Section 3.4), and KMNIST (Section 3.5) shows a significant drop in model accuracies[2] for (pre-trained) models that perform reasonable well on each individual dataset but produce significantly worse results on the combined dataset (Figure 1.1). This is surprising, because, from a human perspective, all characters seem to be distinguishable easily, except for very few classes like "O","o","0" or "L","l","1". Further, we could assume that a model that initially was trained for 1000 classes of relative small RGB images should be able to classify 46 different symbols that don't look a like.

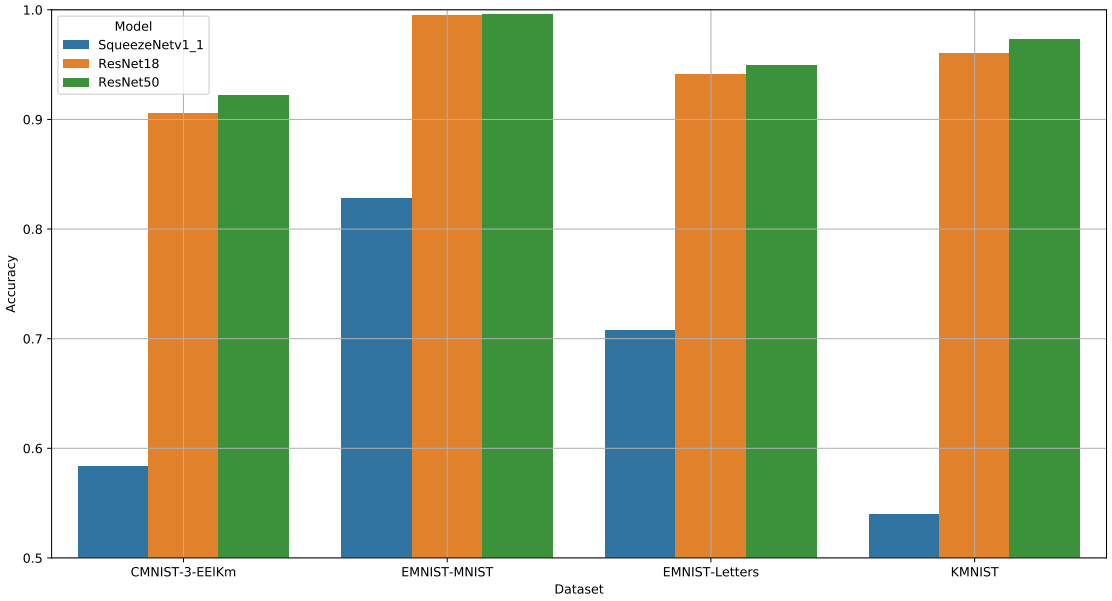


Figure 1.1: Comparison of model accuracies on single datasets and a concatenated version (CMNIST-3-EElKm)

2 The MNIST dilemma

At this point we are not going to discuss if the models we selected are the most suitable for MNIST or not. Academic research and industrial research, if published, needs some public benchmarks for evaluation comparison of new algorithms and model structures. Period. Benchmark datasets vary from discipline to discipline, and a very common benchmark dataset for computer vision is the MNIST dataset. It almost doesn't matter that it hardly represents a real and unsolved computer vision challenge. The error rates of the best models are so low that it is difficult to distinguish them from human performance. Similarly, deep learning has lead to human level performance on many other benchmark datasets. We reached a point at which it seems almost impossible to develop new architectures and algorithms that perform bad on MNIST or other common benchmarks. As mentioned before, we need some form of common benchmark dataset

[1] This dataset became CMNIST-3-EElKm.
[2] Accuracy is not weighted for class imbalances.

to compare algorithms. Therefore, drop-in replacements, such as Fashion-MNIST (Section 3.7), have been published to allow for such a direct comparison, but on a slightly more challenging dataset. This led to a phenomena that both MNIST and Fashion-MNIST benchmark results are provided for new architectures among other standard benchmarks.

Besides all disadvantages of a toy dataset, it provides a good basis for developing new algorithms because MNIST itself and MNIST-like datasets are relatively easy to understand from a human perspective and therefore make algorithm development debug-able. This makes it a lot easier to develop new algorithms that are more sophisticated. A recent example are Capsule Neural Networks which are able to separate overlapping digits [12, 13].

Therefore, we have to ask ourselves why we shouldn't use the benefits of MNIST-like datasets and combine them. Concatenating these datasets leads to more challenging benchmarks while preserving the benefits of using MNIST.

3 Existing MNIST-like datasets

Before we start concatenating datasets, we have to have a look at datasets available and decide if it makes sense to include them in CMNIST subsets or not.

3.1 MNIST

The original, well-known MNIST dataset, which resulted in LeNet-5[14], originates from the NIST (National Institute of Standards and Technology) Special Database 19. Single digits from scans were pre-processed into images of 28x28 pixels. The original dataset provides 6000 training examples per class and 1000 test examples per class. To achieve better representation learning, instead of memorization, a MNIST version with reduced training sets was introduced in 2017 [15]. It is called Reduced MNIST (RMNIST)[3] and provides training sets of sizes 1,5, and 10 per class. We are not going to include MNIST into the CMNIST dataset because we are going to use the full EMNIST dataset which provides a MNIST-like subset anyhow.

3.2 QMNIST

Recently, a proper "forensic analysis" was carried out to find missing samples from the original dataset that were excluded from MNIST to reduce dataset size to match computational requirements back in the days. Besides introducing a few more samples that were not used in MNIST, the QMNIST paper [2] reveals a lot about the generation of MNIST[4] Further, the authors figured out that MNIST-trained algorithms are likely to overfit to the MNIST test set. They observed a drop in accuracy of $< 1\%$. However, we have to ask ourselves if this matters at all. Does it solve any real problems? Can we achieve better representation learning by adding a few more digits to the test set? Since we include EMNIST, we are going to provide more digit examples anyhow and therefore are not going to include QMNIST in CMNIST dataset.

3.3 notMNIST

notMNIST [16] is a dataset that was generated from existing fonts. The letters A to J were used instead of the digits 0 to 9 (Figure 3.1). It was intended as a more challenging MNIST replacement. However, the dataset is a bit larger than MNIST. Therefore, we are going to use a reduced set of notMNIST[5] for CMNIST.

[3] `https://github.com/mnielsen/rmnist`
[4] The "forensic code" is available on GitHub: `https://github.com/facebookresearch/qmnist`.
[5] `https://github.com/davidflanagan/notMNIST-to-MNIST`

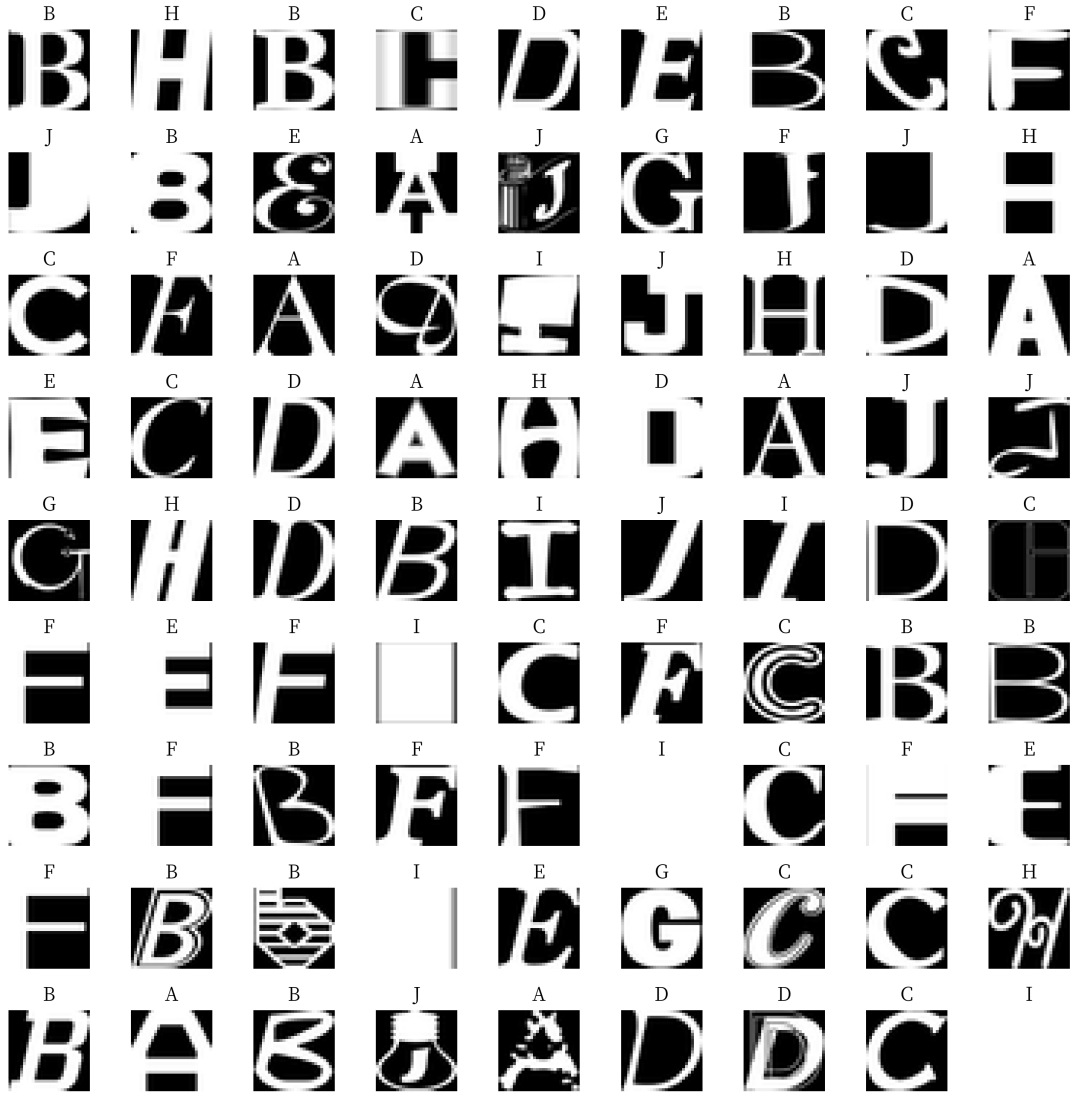


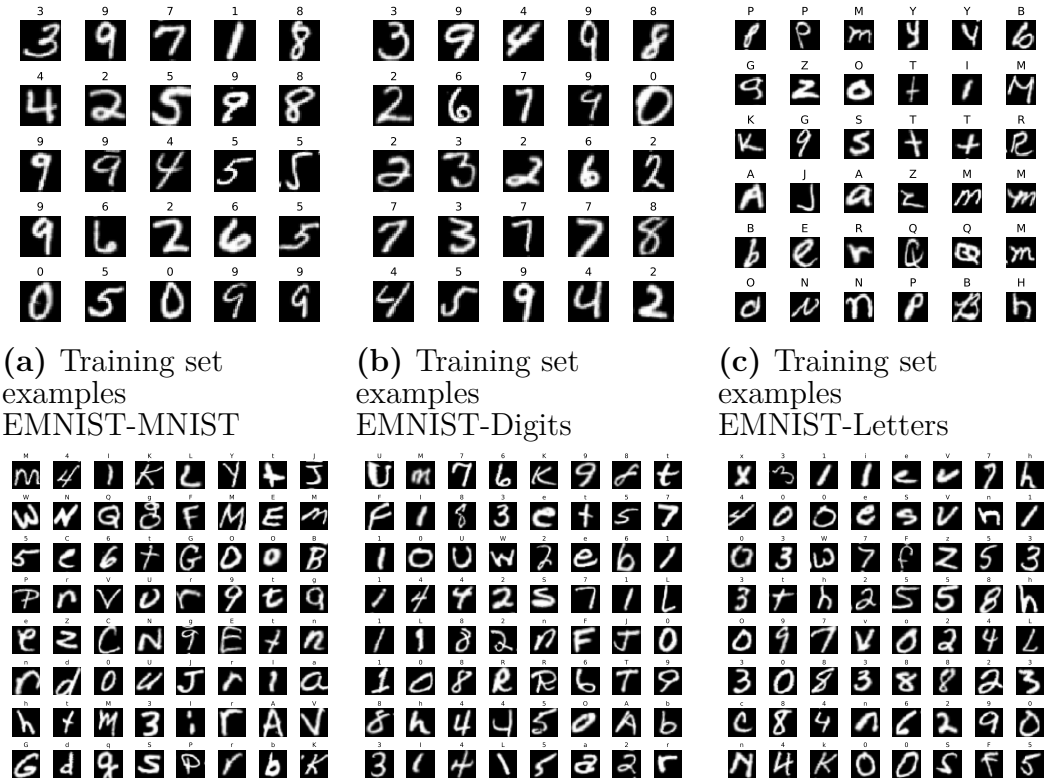
Figure 3.1: Example training images from notMNIST

3.4 EMNIST

Extended MNIST (EMNIST) [17] is a more serious replacement for MNIST. It contains six subsets (Table 3.1) derived from the same database as MNIST. These subsets include letters and, depending on the subset, upper and lower case letters were separated to generate different classes (Figure 3.2). This introduces a new level of complexity for MNIST-like performance evaluation. Furthermore, it introduces unbalanced classes to make things a bit more challenging, even for potential humans in a feedback loop. For example, it is unclear if the image in the lower left corner in Figure 3.2f really belongs to class “n”, or if it belongs to another class (e.g. “N”).

Table 3.1: EMNIST subsets

Subset	No. classes	Classes	Train set size	Test set size	Balanced classes
MNIST	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	60k	10k	True
Digits	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	240k	40k	True
Letters	26	A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z	88.8k	14.8k	True
Balanced	47	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, d, e, f, g, h, n, q, r, t	112.8k	18.8k	True
By-merge	47	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, d, e, f, g, h, n, q, r, t	697932	116323	False
By-class	62	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z	697932	116323	False



(a) Training set examples EMNIST-MNIST

(b) Training set examples EMNIST-Digits

(c) Training set examples EMNIST-Letters

(d) Training set examples EMNIST-Balanced

(e) Training set examples EMNIST-By-merge

(f) Training set examples EMNIST-By-class

Figure 3.2: Example training images from all six EMNIST subsets

3.5 Kuzushiji-MNIST

The Kuzushiji dataset is a drop-in replacement for the MNIST dataset that consists of historic Japanese letters [18]. The target variables are modern days equivalents of these letters. The Kuzushiji dataset consists of three subsets:

- Kuzushiji-MNIST (KMNIST)
- Kuzushiji-49 (K49)
- Kusuzhiji-Kanji.

KMNIST consists of 10 classes and has a similar setup as MNIST (Figure 3.3). Both KMNIST and K49 contain Hiragana letters, Kuzushiji-Kanji contains Kanji letters. Kuzushiji-49 is a highly imbalanced dataset, whereas KMNIST consists of 10 balanced classes. Unlike KMNIST and K49, Kuzushiji-Kanji is highly imbalanced and not pre-splitted into train and test since it contains some classes with a single example only. Therefore, we include KMNIST and K49 only.

Figure 3.3: Example training images from Kuzushiji-KMNIST

3.6 Kannada-MNIST

Kannada-MNIST is another non-western symbols MNIST dataset [19]. It is a dataset that contains symbols from the Kannada language which is the official language in the state of Karnataka in India. It follows the MNIST layout and therefore provides 10 classes with 6000 training examples each (Figure 3.4). It comes with two test sets, however, we are going to include the Kannada-MNIST test set only.

3.7 Fashion-MNIST

Fashion-MNIST is another drop-in replacement for MNIST [20]. It originates from Zalando research and contains 10 classes of different clothing images (Figure 3.5). Each sample is a gray scale image in MNIST size. A public benchmark services is offered by the authors of this dataset [3].

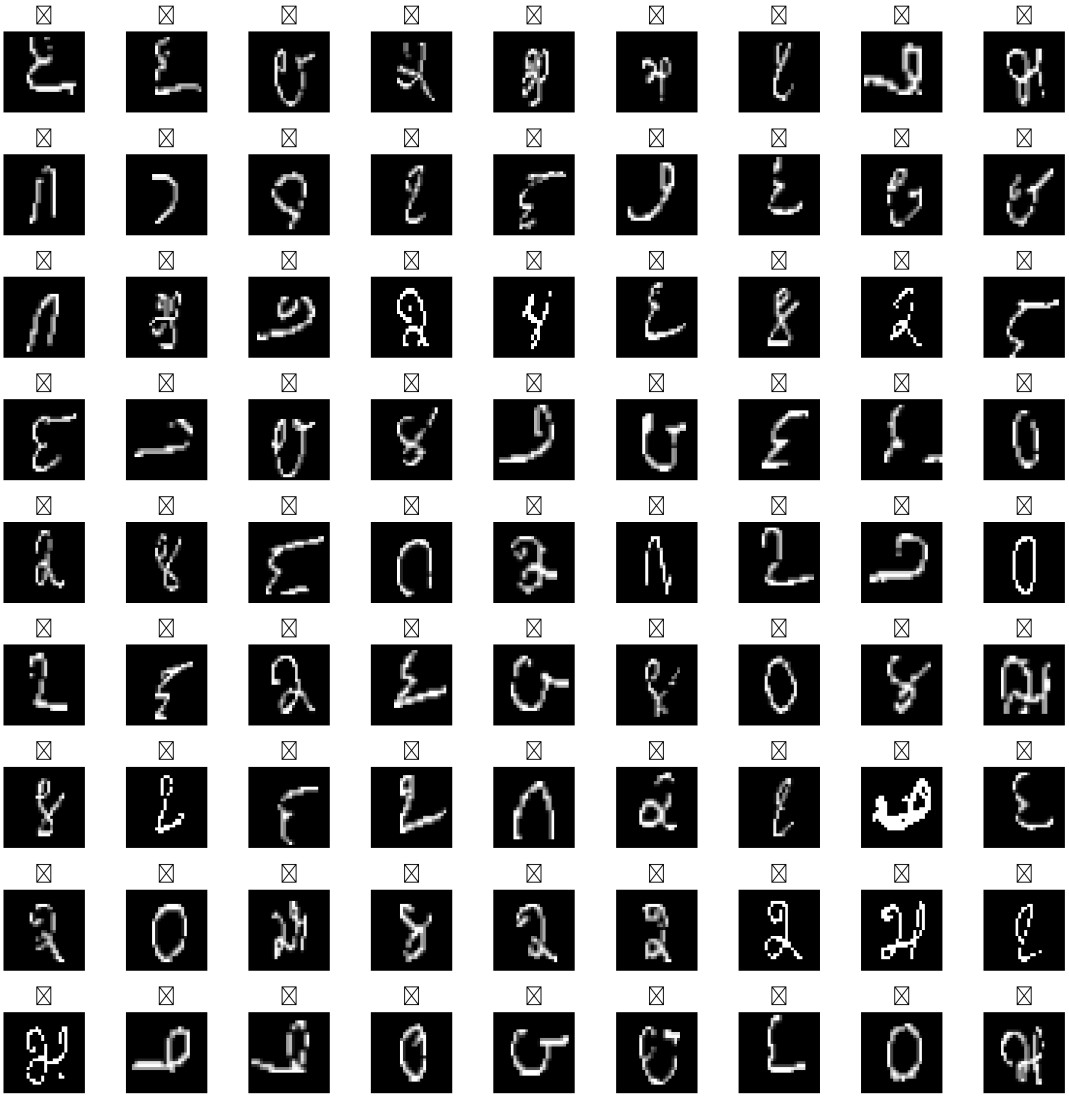


Figure 3.4: Example training images from Kannada-MNIST

3.8 Honorable mentions

Many other datasets were published that use some kind of MNIST-like image format. A prominent example is Quick, Draw! by Google. Quick, Draw! is a dataset of quick drawings (doodles). The trace of the mouse pointer is recorded. However, MNIST-like dumps are offered[6] as well.

Another interesting example is the HASYv2 dataset [21]. It contains 168233 samples of 369 classes of different handwritten (mathematical) symbols. Unlike other MNIST-like datasets, it utilizes an image size of 32x32.

MultiMNIST is an approach to create overlapping digits within the limitations of MNIST using digits from MNIST. This dataset was generated and used to demonstrate that capsule neural networks can separate individual, overlapping digits and classify them correctly [12].

[6] `https://github.com/googlecreativelab/quickdraw-dataset/issues/19`

Figure 3.5: Example training images from Fashion-MNIST

4 Introducing CMNIST

With CMNIST, we don't provide any new data to the research community. However, we concatenate existing datasets to create more challenging datasets[7]. The licenses of the original datasets and the original datasets themselves remain unchanged.

4.1 CMNIST structure

CMNIST is structured in a way that it remains expandable. Each MNIST-like dataset used for CMNIST gets an abbreviation (Table 4.1). The final dataset name consists of the prefix "CMNIST-", followed by a number stating how many datasets have been concatenated. The abbreviations follow in alphabetical order afterwards. Therefore, the dataset "CMNIST-2-EF" is generated from two datasets, namely EMNIST-MNIST and Fashion-MNIST.

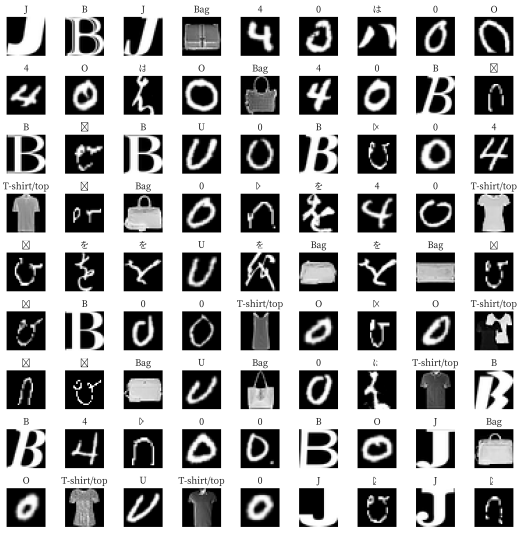
[7] CMNIST generator script available on GitLab: `https://gitlab.com/simonwenkel/cmnist`.

Combining these datasets without overlapping classes, results in 134 CMNIST subsets (Table 4.2). Despite offering a variety of extendable dataset combinations, we provide a baseline dataset. CMNIST-X is a reference dataset that originally was generated by random selection of classes across all MNIST-like datasets used to generate CMNIST datasets. We sampled 2 and 4 classes from EMNIST-MNIST, EMNIST-Letters, fashionMNIST, KMNIST and notMNIST, resulting in the CMNIST-X-12 (Figure 4.1a) and CMNIST-X-24 subsets (Figure 4.1b).

Similar to RMNIST (reduced MNIST), we propose to used reduced training sets as well. The test sets remain untouched. Reduced CMNIST subsets are named using the convention CMNIST-{}-{}-r-{train sample per class}.

Table 4.1: Subset nomenclature for concatenating MNIST-like datasets

Dataset	Subset	Abbreviation
EMNIST	MNIST	E
EMNIST	Digits	Ed
EMNIST	Letters	El
EMNIST	Balanced	Eb
EMNIST	By-merge	Em
EMNIST	By-class	Ec
Fashion-MNIST	-	F
Kannada-MNIST	-	Kd
Kuzushiji	KMNIST	Km
Kuzushiji	K49	K49
notMNIST	-	N



(a) CMNIST-X-12 examples

(b) CMNIST-X-24 examples

Figure 4.1: Example training images from CMNIST-X sets

Table 4.2: Overview of CMNIST subsets

Subset	Balanced	No. classes	Subset	Balanced	No. classes
CMNIST-2-EEl	No	36	CMNIST-2-EF	Yes	20
CMNIST-2-EK49	No	59	CMNIST-2-EKd	Yes	20
CMNIST-2-EKm	Yes	20	CMNIST-2-EN	Yes	20
CMNIST-2-EbF	No	57	CMNIST-2-EbK49	No	96
CMNIST-2-EbKd	No	57	CMNIST-2-EbKm	No	57
CMNIST-2-EcF	No	72	CMNIST-2-EcK49	No	111
CMNIST-2-EcKd	No	72	CMNIST-2-EcKm	No	72
CMNIST-2-EdEl	No	36	CMNIST-2-EdF	No	20
CMNIST-2-EdK49	No	59	CMNIST-2-EdKd	No	20
CMNIST-2-EdKm	No	20	CMNIST-2-EdN	No	20
CMNIST-2-ElF	No	36	CMNIST-2-ElK49	No	75
CMNIST-2-ElKd	No	36	CMNIST-2-ElKm	No	36
CMNIST-2-EmF	No	57	CMNIST-2-EmK49	No	96
CMNIST-2-EmKd	No	57	CMNIST-2-EmKm	No	57
CMNIST-2-FK49	No	59	CMNIST-2-FKd	Yes	20
CMNIST-2-FKm	Yes	20	CMNIST-2-FN	Yes	20
CMNIST-2-K49N	No	59	CMNIST-2-KdK49	No	59
CMNIST-2-KdKm	Yes	20	CMNIST-2-KdN	Yes	20
CMNIST-2-KmN	Yes	20	CMNIST-3-EElF	No	46
CMNIST-3-EElK49	No	85	CMNIST-3-EElKd	No	46
CMNIST-3-EElKm	No	46	CMNIST-3-EFK49	No	69
CMNIST-3-EFKd	Yes	30	CMNIST-3-EFKm	Yes	30
CMNIST-3-EFN	Yes	30	CMNIST-3-EK49N	No	69
CMNIST-3-EKdK49	No	69	CMNIST-3-EKdKm	Yes	30
CMNIST-3-EKdN	Yes	30	CMNIST-3-EKmN	Yes	30
CMNIST-3-EbFK49	No	106	CMNIST-3-EbFKd	No	67
CMNIST-3-EbFKm	No	67	CMNIST-3-EbKdK49	No	106
CMNIST-3-EbKdKm	No	67	CMNIST-3-EcFK49	No	121
CMNIST-3-EcFKd	No	82	CMNIST-3-EcFKm	No	82
CMNIST-3-EcKdK49	No	121	CMNIST-3-EcKdKm	No	82
CMNIST-3-EdElF	No	46	CMNIST-3-EdElK49	No	85
CMNIST-3-EdElKd	No	46	CMNIST-3-EdElKm	No	46
CMNIST-3-EdFK49	No	69	CMNIST-3-EdFKd	No	30
CMNIST-3-EdFKm	No	30	CMNIST-3-EdFN	No	30
CMNIST-3-EdK49N	No	69	CMNIST-3-EdKdK49	No	69
CMNIST-3-EdKdKm	No	30	CMNIST-3-EdKdN	No	30
CMNIST-3-EdKmN	No	30	CMNIST-3-ElFK49	No	85
CMNIST-3-ElFKd	No	46	CMNIST-3-ElFKm	No	46
CMNIST-3-ElKdK49	No	85	CMNIST-3-ElKdKm	No	46
CMNIST-3-EmFK49	No	106	CMNIST-3-EmFKd	No	67
CMNIST-3-EmFKm	No	67	CMNIST-3-EmKdK49	No	106
CMNIST-3-EmKdKm	No	67	CMNIST-3-FK49N	No	69
CMNIST-3-FKdK49	No	69	CMNIST-3-FKdKm	Yes	30
CMNIST-3-FKdN	Yes	30	CMNIST-3-FKmN	Yes	30
CMNIST-3-KdK49N	No	69	CMNIST-3-KdKmN	Yes	30
CMNIST-4-EElFK49	No	95	CMNIST-4-EElFKd	No	56
CMNIST-4-EElFKm	No	56	CMNIST-4-EElKdK49	No	95
CMNIST-4-EElKdKm	No	56	CMNIST-4-EFK49N	No	79
CMNIST-4-EFKdK49	No	79	CMNIST-4-EFKdKm	Yes	40
CMNIST-4-EFKdN	Yes	40	CMNIST-4-EFKmN	Yes	40
CMNIST-4-EKdK49N	No	79	CMNIST-4-EKdKmN	Yes	40
CMNIST-4-EbFKdK49	No	116	CMNIST-4-EbFKdKm	No	77
CMNIST-4-EcFKdK49	No	131	CMNIST-4-EcFKdKm	No	92
CMNIST-4-EdElFK49	No	95	CMNIST-4-EdElFKd	No	56
CMNIST-4-EdElFKm	No	56	CMNIST-4-EdElKdK49	No	95
CMNIST-4-EdElKdKm	No	56	CMNIST-4-EdFK49N	No	79
CMNIST-4-EdFKdK49	No	79	CMNIST-4-EdFKdKm	No	40
CMNIST-4-EdFKdN	No	40	CMNIST-4-EdFKmN	No	40
CMNIST-4-EdKdK49N	No	79	CMNIST-4-EdKdKmN	No	40
CMNIST-4-ElFKdK49	No	95	CMNIST-4-ElFKdKm	No	56
CMNIST-4-EmFKdK49	No	116	CMNIST-4-EmFKdKm	No	77
CMNIST-4-FKdK49N	No	79	CMNIST-4-FKdKmN	Yes	40
CMNIST-5-EElFKdK49	No	105	CMNIST-5-EElFKdKm	No	66
CMNIST-5-EFKdK49N	No	89	CMNIST-5-EFKdKmN	Yes	50
CMNIST-5-EdElFKdK49	No	105	CMNIST-5-EdElFKdKm	No	66
CMNIST-5-EdFKdK49N	No	89	CMNIST-5-EdFKdKmN	No	50
CMNIST-X-12	No	12	CMNIST-X-24	No	24

5 Baseline results

A major problem in machine learning research remains reproducibility of results. Even little things like not defining random states can lead to slightly different results. Therefore, we use the fastai framework[8] with the following settings:

```
# all random states initiated with  ``seed = 1''
ds_tfms = get_transforms(do_flip=False, flip_vert=False)
data = ImageDataBunch.from_df("", train_df, ds_tfms=ds_tfms, size=28, bs=1024)
learn = cnn_learner(data, model, metrics=accuracy)
learn.lr_find(start_lr=1e-6, end_lr=1e1, stop_div=True, num_it=100)
lr = learn.recorder.min_grad_lr
learn.fit_one_cycle(20, lr)
```

[8] https://github.com/fastai/fastai

We used ResNet18, ResNet50, and SqueezeNet v1.1 models pre-trained on ImageNet. Additionally, we tested tested simple k-Nearest Neighbors as an additional baseline result on the reduced training datasets (CMNIST-r-) (Figures 5.1,5.2).

The reduced CMNIST subsets show a lot of interesting and somehow seemingly weird results for different models. On the CMNIST subsets that use full training sets, we can observe, that the accuracy of ResNet50 is mostly slightly higher than ResNet18 and that using ResNet18 leads to much better results than SqueezeNet (Figure 5.3). However, the performance on the reduced datasets is different. While kNN seems to be the best choice in most cases, the performance of the neural networks is somewhat less clear (Figures 5.1,5.2). SqueezeNet seem to perform better than ResNet with small datasets.

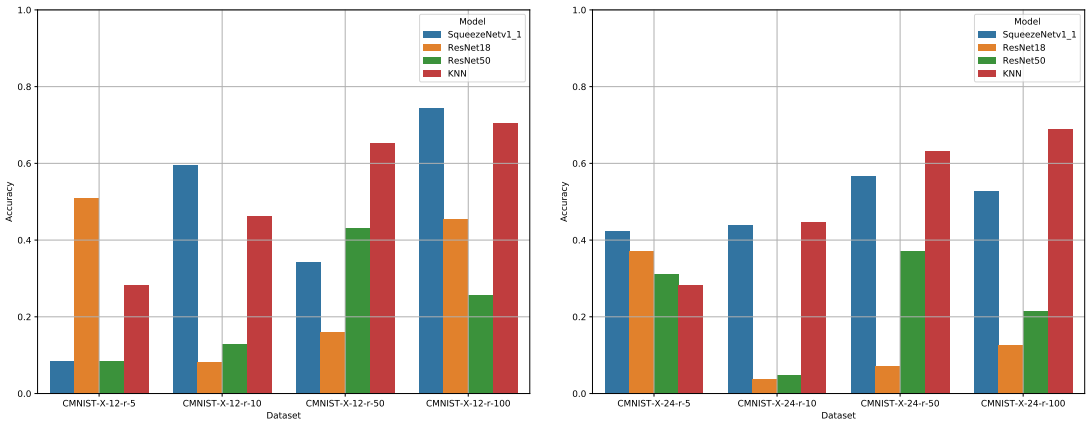


Figure 5.1: Example training images from all six EMNIST subsets

6 Discussion and Conclusions

We propose a new kind of dataset that is not another MNIST dataset but but provides significantly more challenging subsets. However, we have to remind ourselves that once the class count increases, the accuracies will increase because a rather small subset of difficult classes doesn't influence the overall accuracy so much. The results of models pretrained on ImageNet, which has >20000 classes - or 1000 in its reduced form, is less satisfactory. Even a rather simple extension of MNIST results in a significant loss of accuracy compared to a standard MNIST problem. Therefore, we could assume that widely deployed neural network architectures are less suitable for representation learning. This could explain sensitivities to textures and therefore explain the simplicity of various adversarial attacks. With CMNIST, we hope to provide a better basis for research of new neural network architectures such as capsule networks or even biological neural networks (e.g. Moth Olfactory Networks [22]).

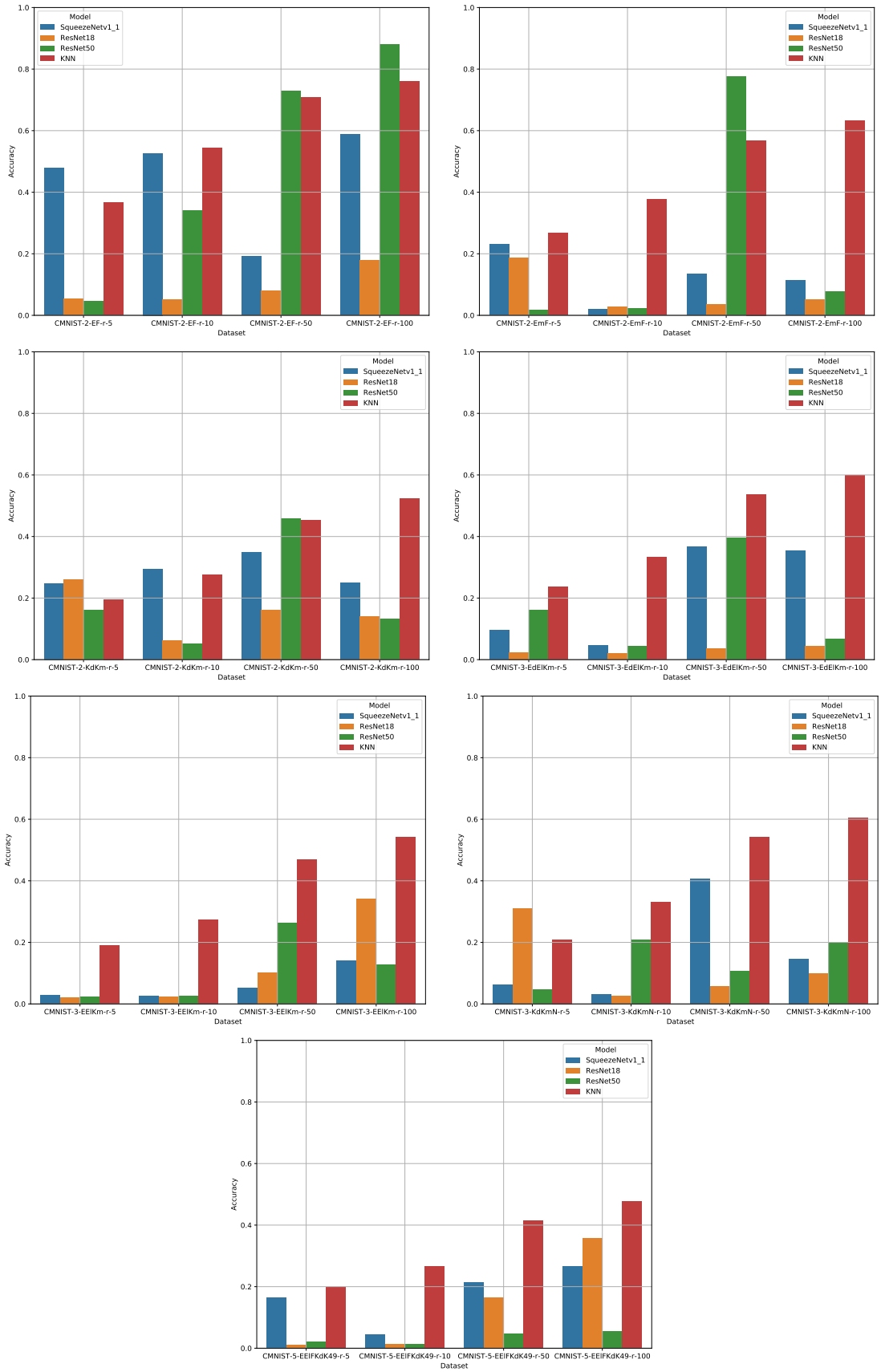


Figure 5.2: Example training images from all six EMNIST subsets

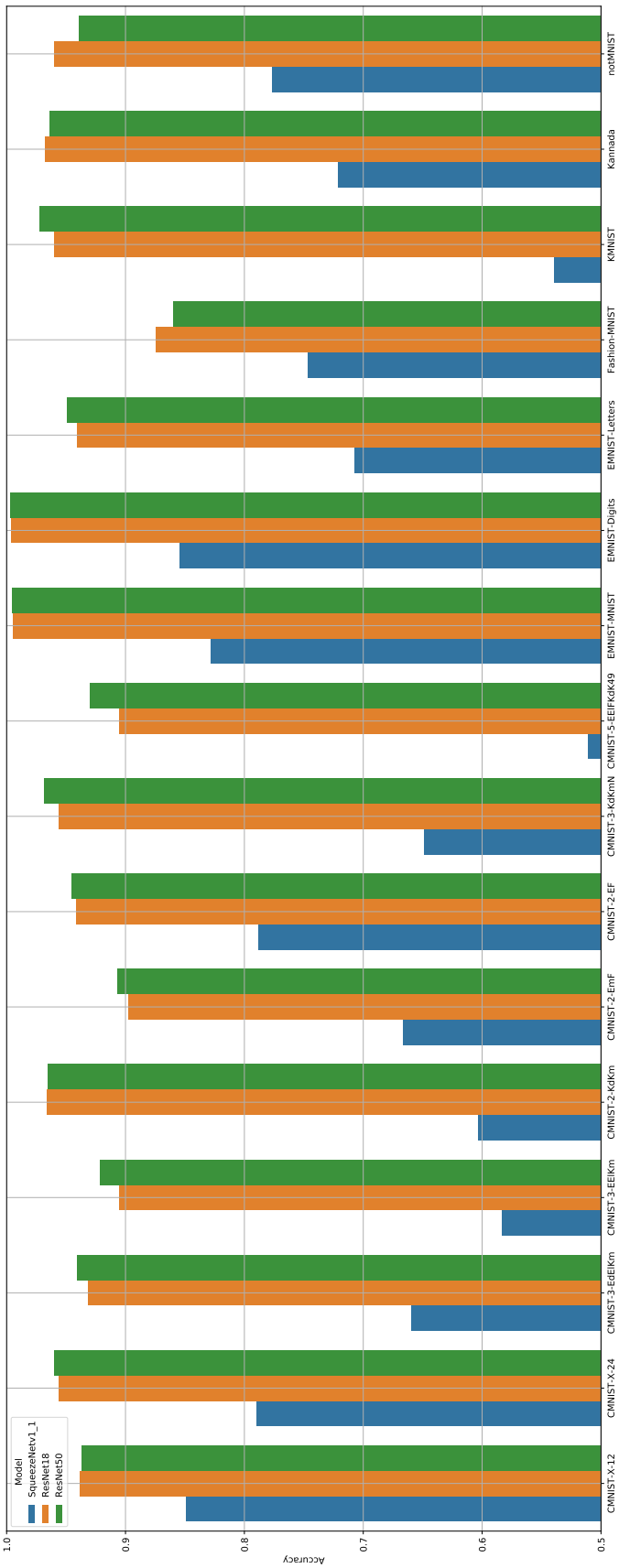


Figure 5.3: Comparison of model accuracies on single datasets and a concatenated version (CMNIST-3-EElKm)

References

[1] Y. LeCun, C. Cortes, and C. J. C. Burges. *THE MNIST DATABASE of handwritten digits*. 1994. URL: http://yann.lecun.com/exdb/mnist/.

[2] C. Yadav and L. Bottou. *Cold Case: The Lost MNIST Digits*. 2019. arXiv: 1905.10498.

[3] Zalando SE. *MNIST and Fashion-MNIST Benchmarks*. 2017. URL: http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/.

[4] B. Hammer. *Popular Datasets Over Time*. 2017. URL: https://www.kaggle.com/benhamner/popular-datasets-over-time?scriptVersionId=1879813.

[5] S. Basu, M. Karki, S. Ganguly, R. DiBiano, S. Mukhopadhyay, S. Gayaka, R. Kannan, and R. Nemani. "Learning Sparse Feature Representations Using Probabilistic Quadtrees and Deep Belief Nets". In: *Neural Processing Letters* 45.3 (Sept. 2016), pp. 855–867. DOI: 10.1007/s11063-016-9556-4.

[6] D. C. Castro, J. Tan, B. Kainz, E. Konukoglu, and B. Glocker. *Morpho-MNIST: Quantitative Assessment and Diagnostics for Representation Learning*. 2018. arXiv: 1809.10780v2.

[7] G. W. Ding, K. Y. C. Lui, X. Jin, L. Wang, and R. Huang. *On the Sensitivity of Adversarial Robustness to Input Data Distributions*. 2019. arXiv: 1902.08336v1.

[8] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. *Do CIFAR-10 Classifiers Generalize to CIFAR-10?* 2018. arXiv: 1806.00451v1.

[9] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. *Do ImageNet Classifiers Generalize to ImageNet?* 2019. arXiv: 1902.10811v2.

[10] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness." In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=Bygh9j09KX.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[12] S. Sabour, N. Frosst, and G. E. Hinton. "Dynamic Routing Between Capsules". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 3856–3866. URL: http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf.

[13] G. E. Hinton, S. Sabour, and N. Frosst. "Matrix capsules with EM routing". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=HJWLfGWRb.

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[15] M. Nielsen. *Reduced MNIST: how well can machines learn from small data?* 2017. URL: http://cognitivemedium.com/rmnist.

[16] Y. Bulatov. *notMNIST dataset*. 2011. URL: https://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html.

[17] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. *EMNIST: an extension of MNIST to handwritten letters*. 2017. arXiv: 1702.05373v2.

[18] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha. *Deep Learning for Classical Japanese Literature*. 2018. DOI: 10.20676/00000341. arXiv: 1812.01718v1.

[19] V. U. Prabhu. *Kannada-MNIST: A new handwritten digits dataset for the Kannada language*. 2019. arXiv: 1908.01242v1.

[20] H. Xiao, K. Rasul, and R. Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017. arXiv: arXiv:1708.07747.

[21] M. Thoma. *The HASYv2 dataset*. 2017. arXiv: 1701.08380.

[22] C. B. Delahunt and J. N. Kutz. "Putting a bug in ML: The moth olfactory network learns to read MNIST". In: *Neural Networks* 118 (2019), pp. 54–64. DOI: 10.1016/j.neunet.2019.05.012.