A. Broader Impact

The action score proposed in this work is a measure of difficulty for samples in a training or validation/test sets. We expect that our method will be used in practice to measure difficulty of datasets and evaluate model biases, which can provide a more comprehensive evaluation and comparison of machine learning models.

There is much interest in finding model biases, which is very applicable for the fairness and transparency field, to anticipate possible issues when deploying models in realistic situations. This likely requires the use of multiple validation and test sets, as single datasets can still be biased and the action score can only provide difficulty estimations in datasets used for training or evaluation.

We expect that in the long term, our method will influence societal interests by aiding machine learning engineers and researchers to be aware of their model biases before these models are deployed in real-life situations. This would only be possible if engineers and researchers actually use our proposed method, but in any case it adds to the toolbox to debug models.

Our method could also be misused, for example, to show that a model is not biased, by manipulating the training and validation sets and removing difficult samples, which would still produce a biased model, but of course the action score would not show these biases. Our proposed method does not replace basic transparency requirements on a model, such as model cards [18].

B. Model and Training Details

In this section we present all the relevant model details of the training process. We used in all classification models ADAM with a batch size of 32 and initial learning rate of 0.001. We used early stopping with a stop patience of five and learning rate reduction factor 0.1 with patience of two. The only image preprocessing was an image normalization by dividing all images by 255.0.

B.1. Keras-CNN

This model can be compactly represented in the following pseudocode:

```
image = Input(shape=(32,32,3))
x = Conv2D(32, (3, 3), 'relu')(image)
x = Conv2D(64, (3, 3), 'relu')(x)
x = MaxPooling2D((2, 2))(x)
x = Dropout(0.25)(x)
x = Flatten()(x)
x = Dense(128, 'relu')(x)
x = Dropout(0.5)(x)
y = Dense(num_classes, 'softmax')(x)
```

B.2. ResNetV2

We used a ResNetV2 architecture with 31 convolutional layers containing $849,002$ paramters for an input shape of $[32, 32, 3]$. The ResNetV2 architectures uses 3 blocks of batch-normalization $\rightarrow$ relu $\rightarrow$ convolution per residual connection.

B.3. Xception Mini

We used an Xception architecture consisting of a stem base of two convolutional layers with the following filters: $[32, 64]$. Then we apply Xception blocks (residual connections with depth-separable convolutions) using the following filters $[128, 128, 256, 256, 512, 512, 1024]$.

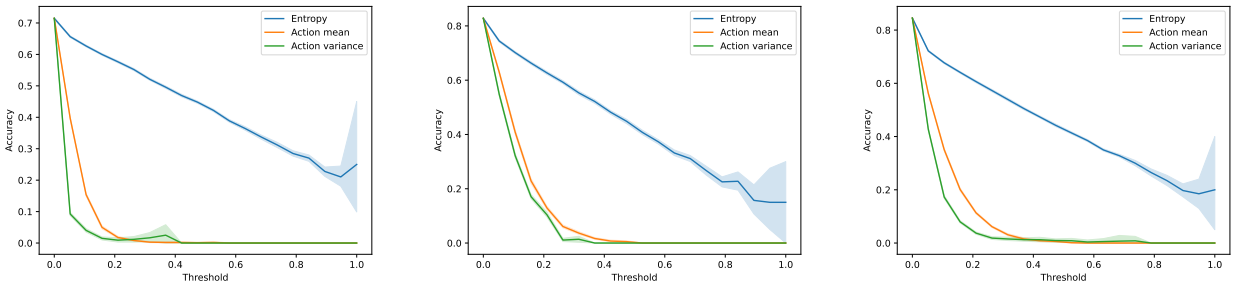
C. Classification Performance of Evaluated Models

In this section we present an evaluation of the test accuracy of the models we trained, as most of our paper is about the evaluation of the action scores and their uses to debug models. These results are shown in Table 2.

Dataset	Keras-CNN	ResNet V2	Mini Xception	Dataset	Keras-CNN	ResNet V2	Mini Xception
MNIST	99.17%	N/A	N/A	FERPlus	69.56%	78.77%	80.55%
CIFAR10	71.39%	83.02%	84.47%	Kuzushiji MNIST	95.31%	N/A	97.95%

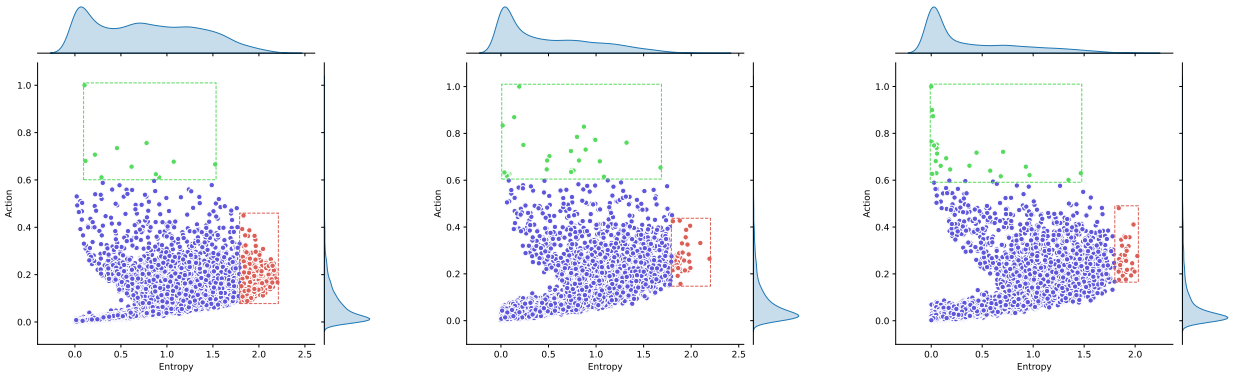
Table 2. Test accuracy of Models we trained to extract and evaluate action scores.

D. Additional Image Classification Results



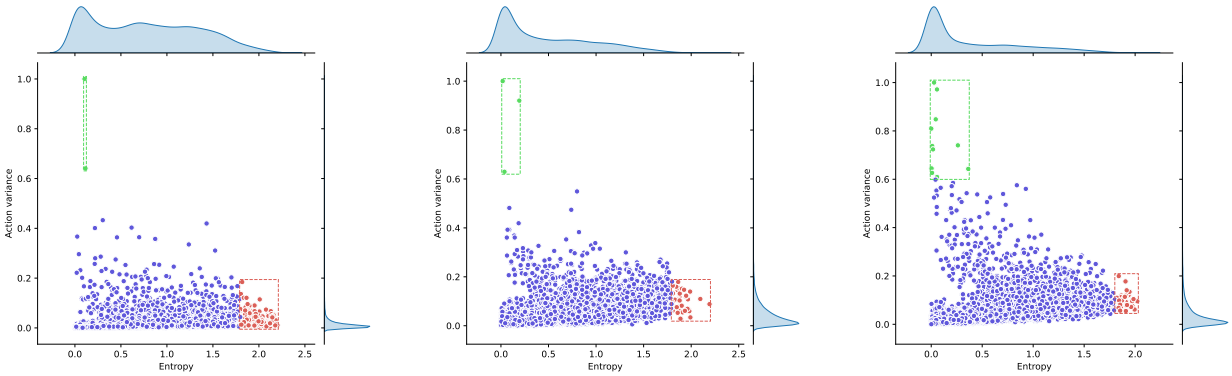
(a) CNN Keras (b) ResNet V2 (c) Xception Mini

Figure 15. Accuracy as a function of metrics for CIFAR10



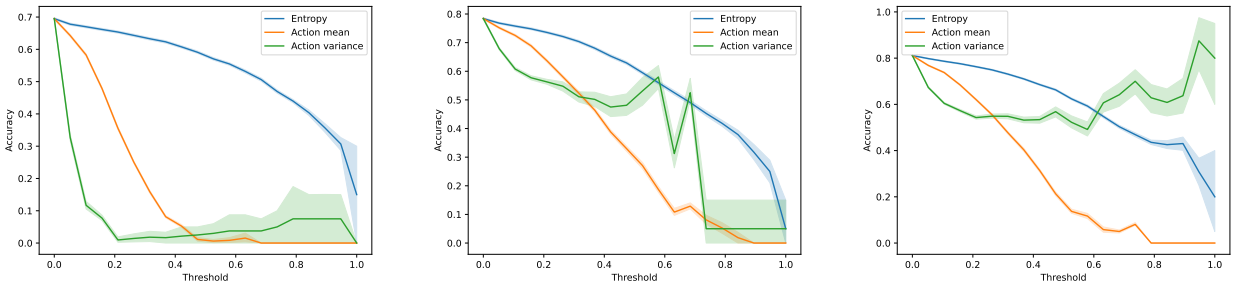
(a) CNN Keras (b) ResNet V2 (c) Xception Mini

Figure 16. Action vs Entropy for multiple models with the CIFAR10 dataset



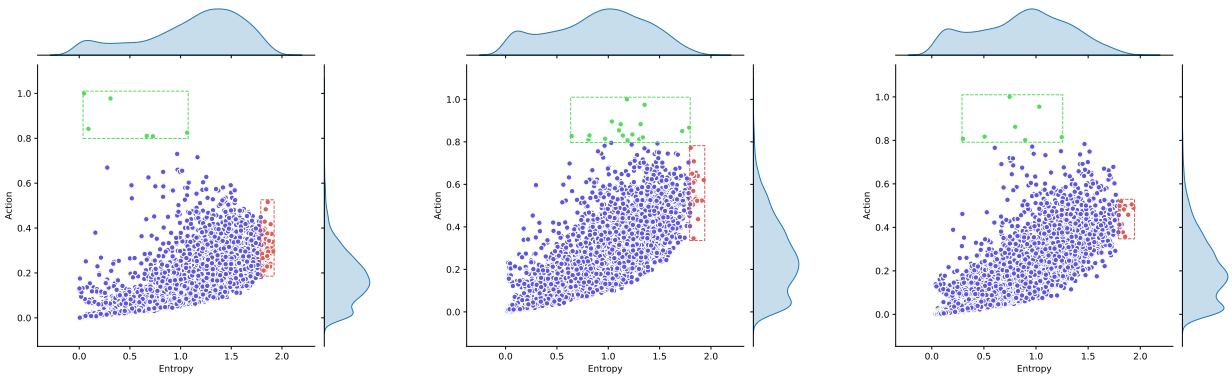
(a) CNN Keras (b) ResNet V2 (c) Xception Mini

Figure 17. Action variance vs Entropy for multiple models with the CIFAR10 dataset



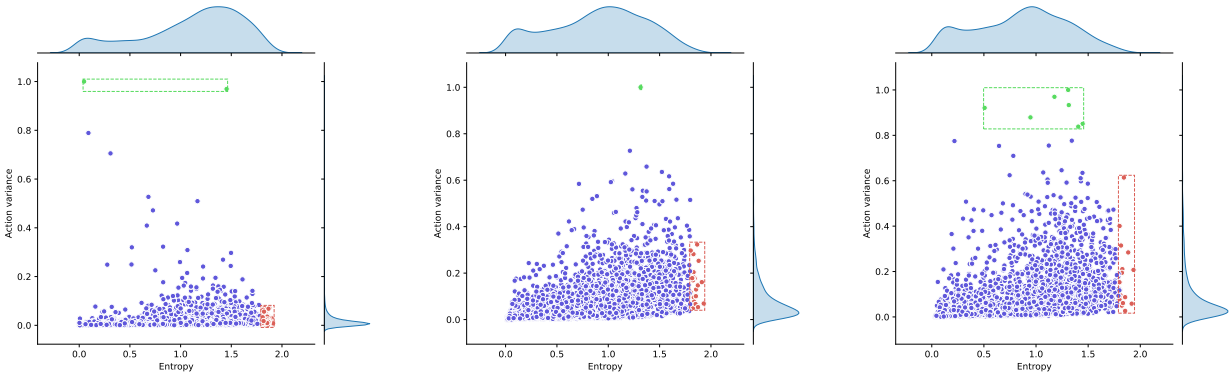
(a) CNN Keras (b) ResNet V2 (c) Xception Mini

Figure 18. Accuracy as a function of metrics for FERPlus



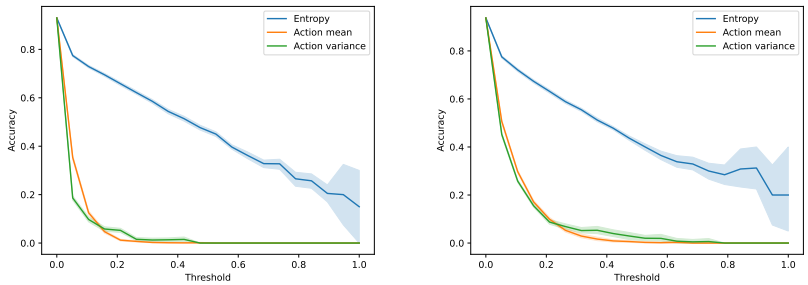
(a) CNN Keras (b) ResNet V2 (c) Xception Mini

Figure 19. Action vs Entropy for multiple models with the FERPlus dataset



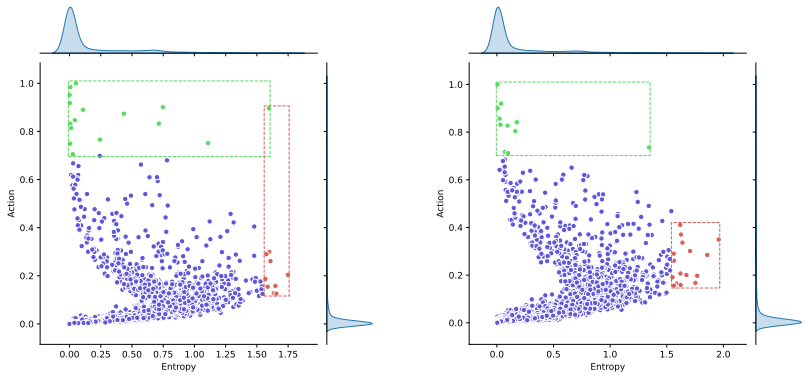
(a) CNN Keras (b) ResNet V2 (c) Xception Mini

Figure 20. Action variance vs Entropy for multiple models with the FERPlus dataset



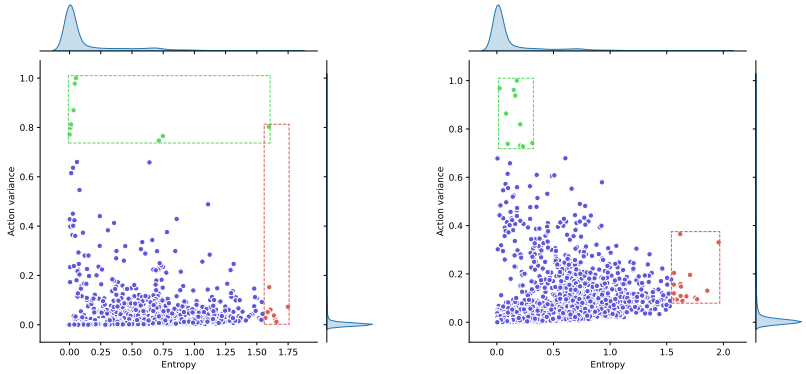
(a) CNN Keras (b) Xception Mini

Figure 21. Accuracy as a function of metrics for FashionMNIST



(a) CNN Keras (b) Xception Mini

Figure 22. Action vs Entropy for multiple models with the FashionMNIST dataset



(a) CNN Keras (b) Xception Mini

Figure 23. Action variance vs Entropy for multiple models with the FashionMNIST dataset

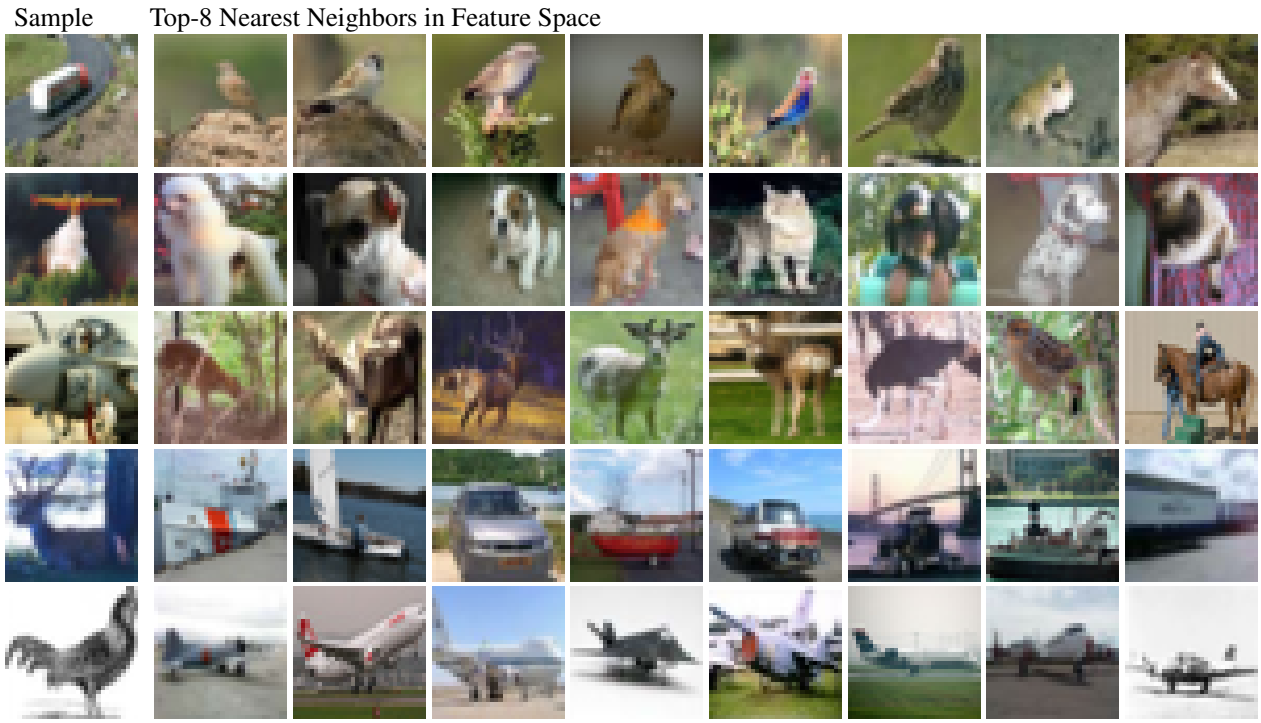


Figure 24. The first element of each row corresponds to one of the top five hardest samples on CIFAR10 test split with a CNN Keras model. The subsequent elements in each row show the top eight neighbors of that first row element.

Figure 25. The first element of each row corresponds to one of the top five hardest samples on CIFAR10 test split with a Xception Mini model. The subsequent elements in each row show the top eight neighbors of that first row element.

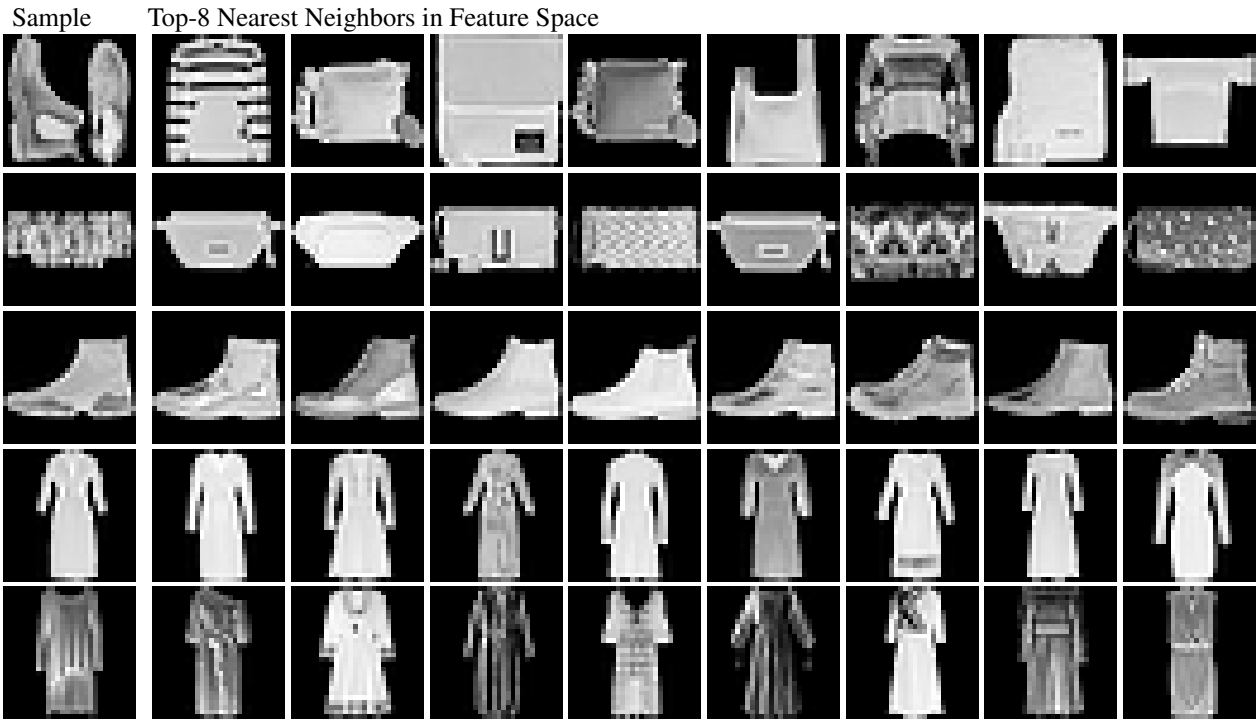


Figure 26. The first element of each row corresponds to one of the top five hardest samples on FashionMNIST test split with a CNN Keras model. The subsequent elements in each row show the top eight neighbors of that first row element.

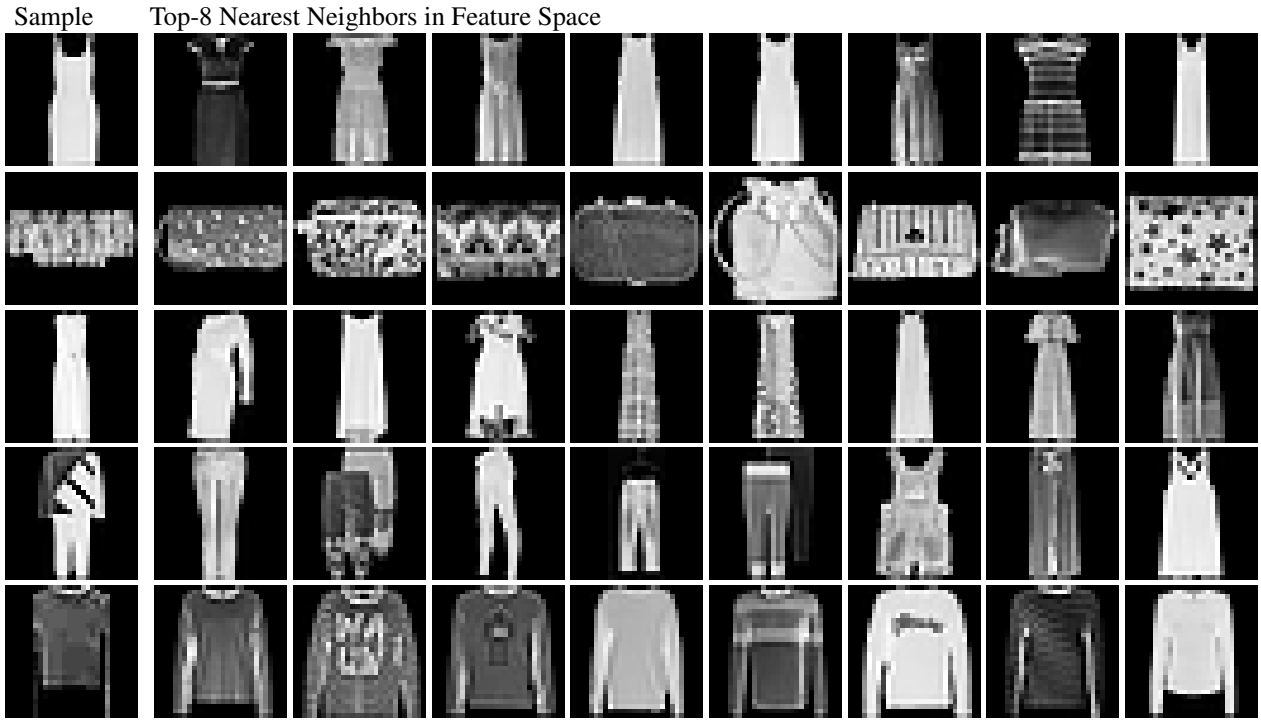


Figure 27. The first element of each row corresponds to one of the top five hardest samples on FashionMNIST test split with a Xception Mini model. The subsequent elements in each row show the top eight neighbors of that first row element.

Figure 28. The first element of each row corresponds to one of the top five hardest samples on FERPlus test split with a CNN Keras model. The subsequent elements in each row show the top eight neighbors of that first row element.

Figure 29. The first element of each row corresponds to one of the top five hardest samples on FERPlus test split with a Xception Mini model. The subsequent elements in each row show the top eight neighbors of that first row element.

Figure 30. The first element of each row corresponds to one of the top five hardest samples on FERPlus test split with a ResNet model. The subsequent elements in each row show the top eight neighbors of that first row element.

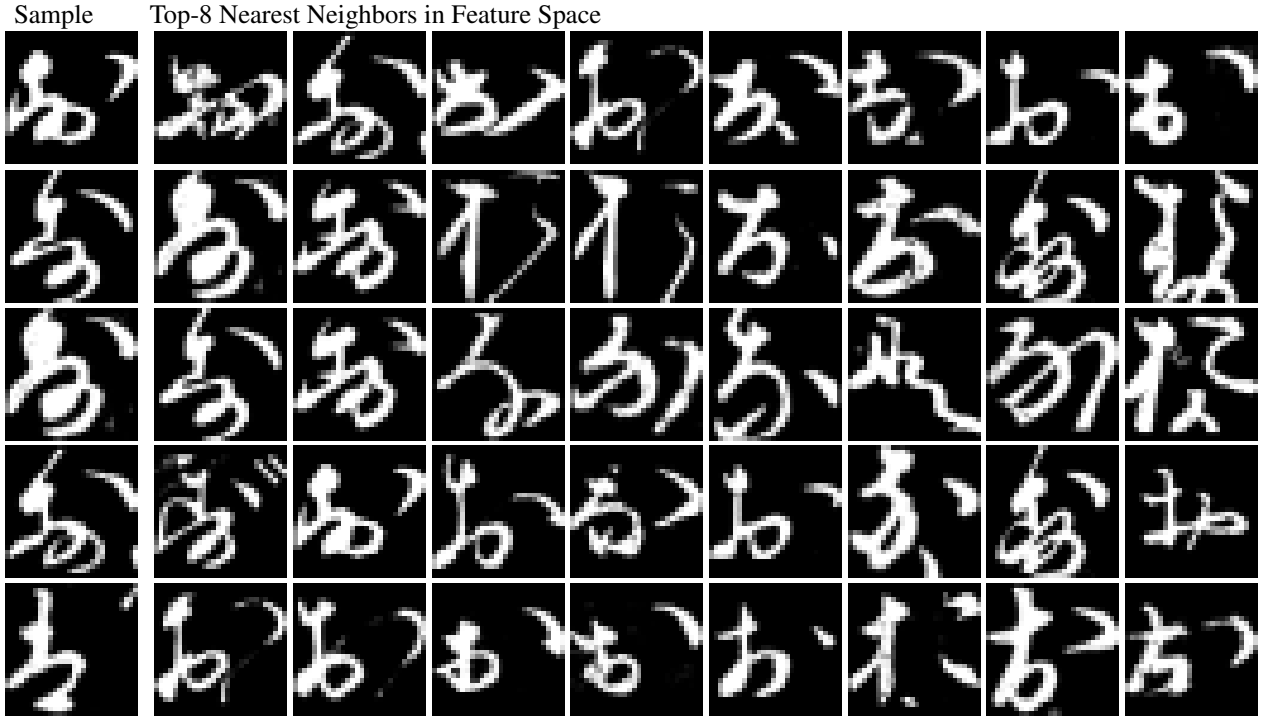


Figure 31. The first element of each row corresponds to one of the top five hardest samples on KuzushijiMNIST test split with a CNN Keras model. The subsequent elements in each row show the top eight neighbors of that first row element.

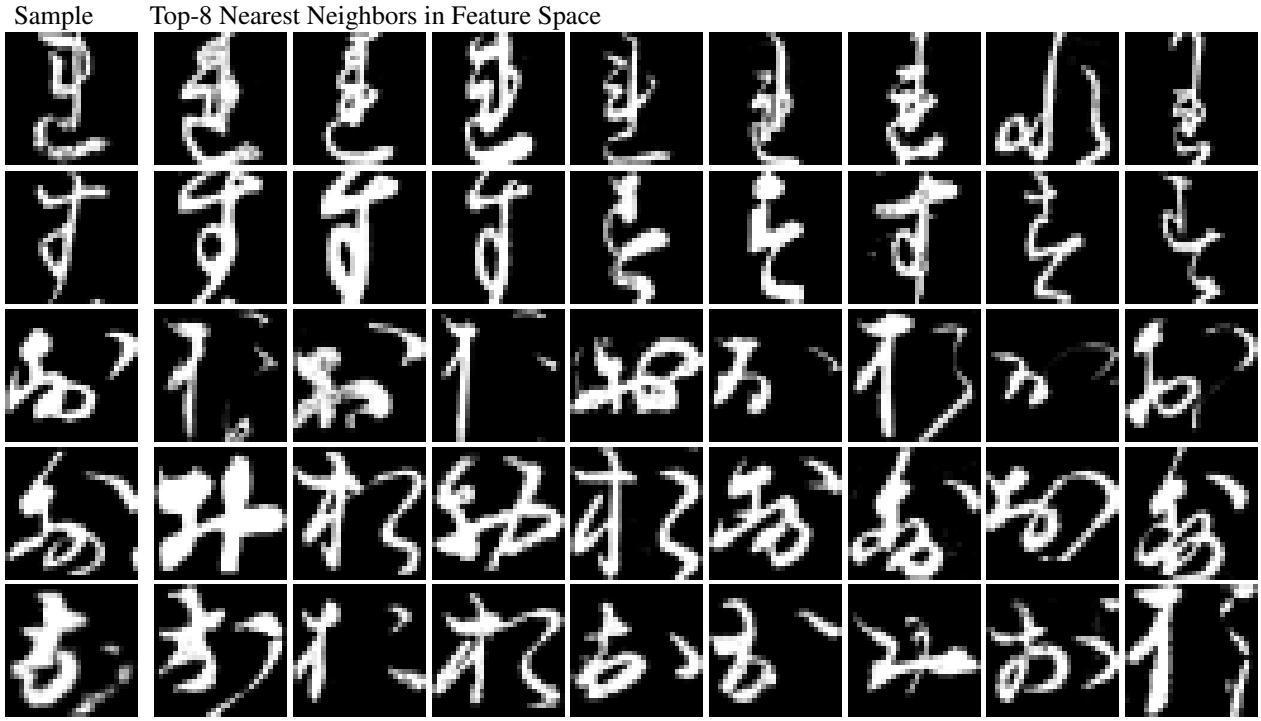


Figure 32. The first element of each row corresponds to one of the top five hardest samples on KuzushijiMNIST test split with a Xception Mini model. The subsequent elements in each row show the top eight neighbors of that first row element.

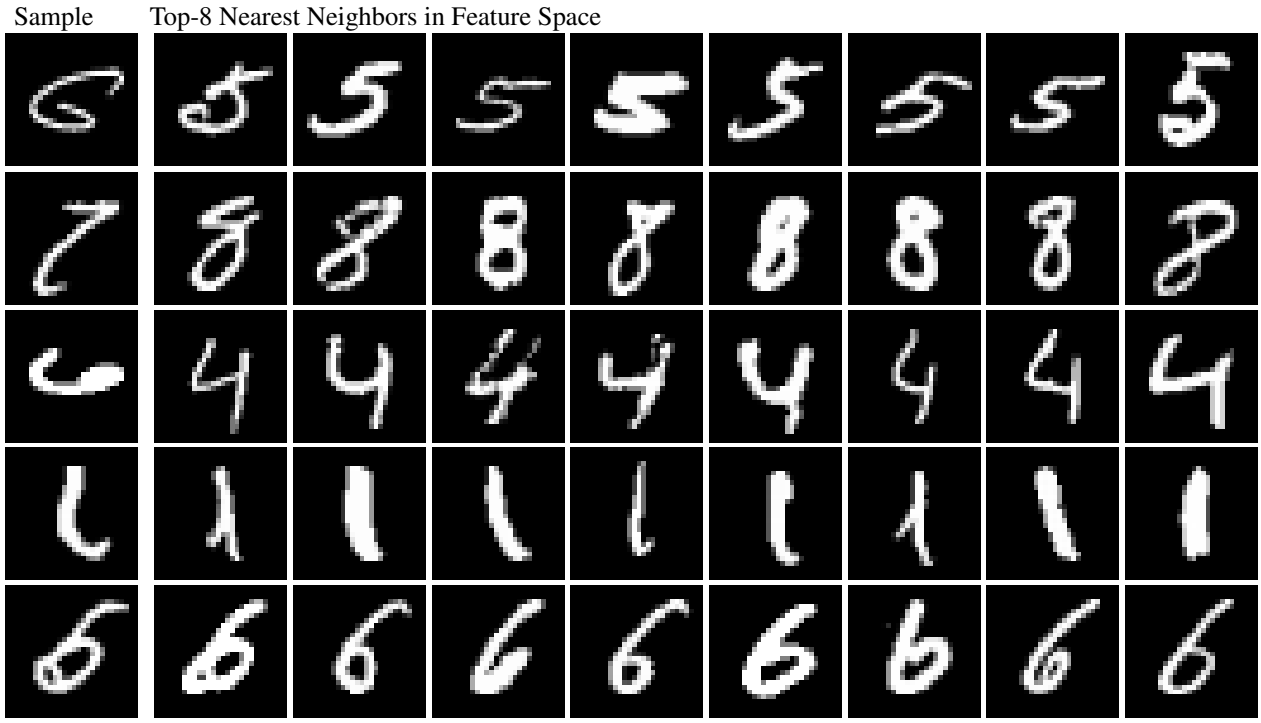


Figure 33. The first element of each row corresponds to one of the top five hardest samples on MNIST test split with a CNN Keras model. The subsequent elements in each row show the top eight neighbors of that first row element.

E. Additional Results on Autoencoders

In this section we present additional results of our method now applied to an unsupervised task. We trained an auto-encoder and computed both the action score per sample and per pixel. We show results in the following three datasets: KuzushijiMNIST, FashionMNIST and CIFAR10. We used the same training parameters specified in our classification experiments. The only difference was the we used a binary-crossentropy loss. The per-pixel difficulty masks shown in Figure 34 and in Figure 35 indicates hardest values with a lighter mask and the easiest ones with darker values. We can observe in the Figure 34 that in the KuzushijiMNIST dataset the most difficult pixels to reconstruct are those around the borders. Furthermore, the hardest images to reconstruct from the FashionMNIST dataset are those that cover the whole image. The hardest pixels to reconstruct in CIFAR10 are those related to the background.

Figure 35 shows the easiest samples. Within all datasets in this Figure we observe that the images with the lowest accumulated loss corresponds to objects that use the least amount non-zero pixels to describe. Moreover, in CIFAR10 those images correspond simple images with a small set of different colors. We believe these results show evidence that our metric does provide an estimation of difficulty in an unsupervised task. The pseudocode of the auto-encoder model used in all of our experiments is presented below.

```
# encoder
x = Conv2D(32, (3, 3), strides=(2, 2), padding='same')(image)
x = Activation('relu', name='relu_1')(x)
x = Conv2D(64, (3, 3), strides=(2, 2), padding='same')(x)
x = Activation('relu', name='relu_2')(x)
x = Conv2D(128, (3, 3), strides=(2, 2), padding='same')(x)
x = Activation('relu', name='relu_3')(x)
x = Conv2D(256, (3, 3), strides=(2, 2), padding='same')(x)
(batch, x_size, y_size, filters) = x.shape
x = Activation('relu')(x)
x = Flatten()(x)
z = Dense(128)(x)
```

```
# decoder
x = Dense(x_size * y_size * filters)(z)
x = Reshape((x_size, y_size, filters))(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(128, (3, 3), padding='same')(x)
x = Activation('relu')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(64, (3, 3), padding='same')(x)
x = Activation('relu')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), padding='same')(x)
x = Activation('relu')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(num_channels, (3, 3), padding='same')(x)
y = Activation('sigmoid')(x)
```

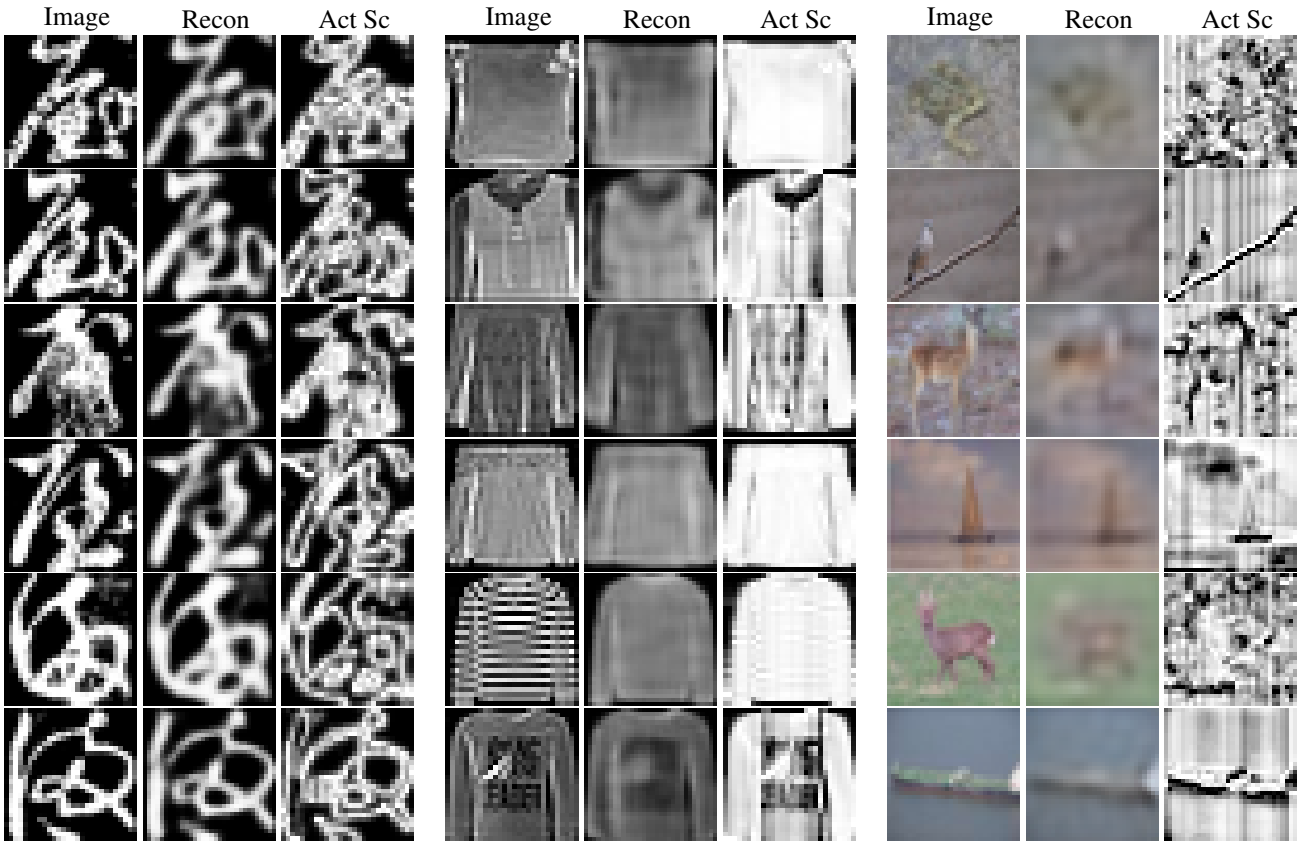


Figure 34. Hardest test samples for KuzushijiMNIST, FashionMNIST and CIFAR10 using our auto-encoder model and our action-score metric. For each dataset we show three columns. The first column from the left corresponds to the real image, the middle one is the reconstruction outputted by our auto-encoder, and the third one is the per-pixel action score indicating which pixels were the hardest (lighter values) to reconstruct, and which ones are the easiest ones (darker values).

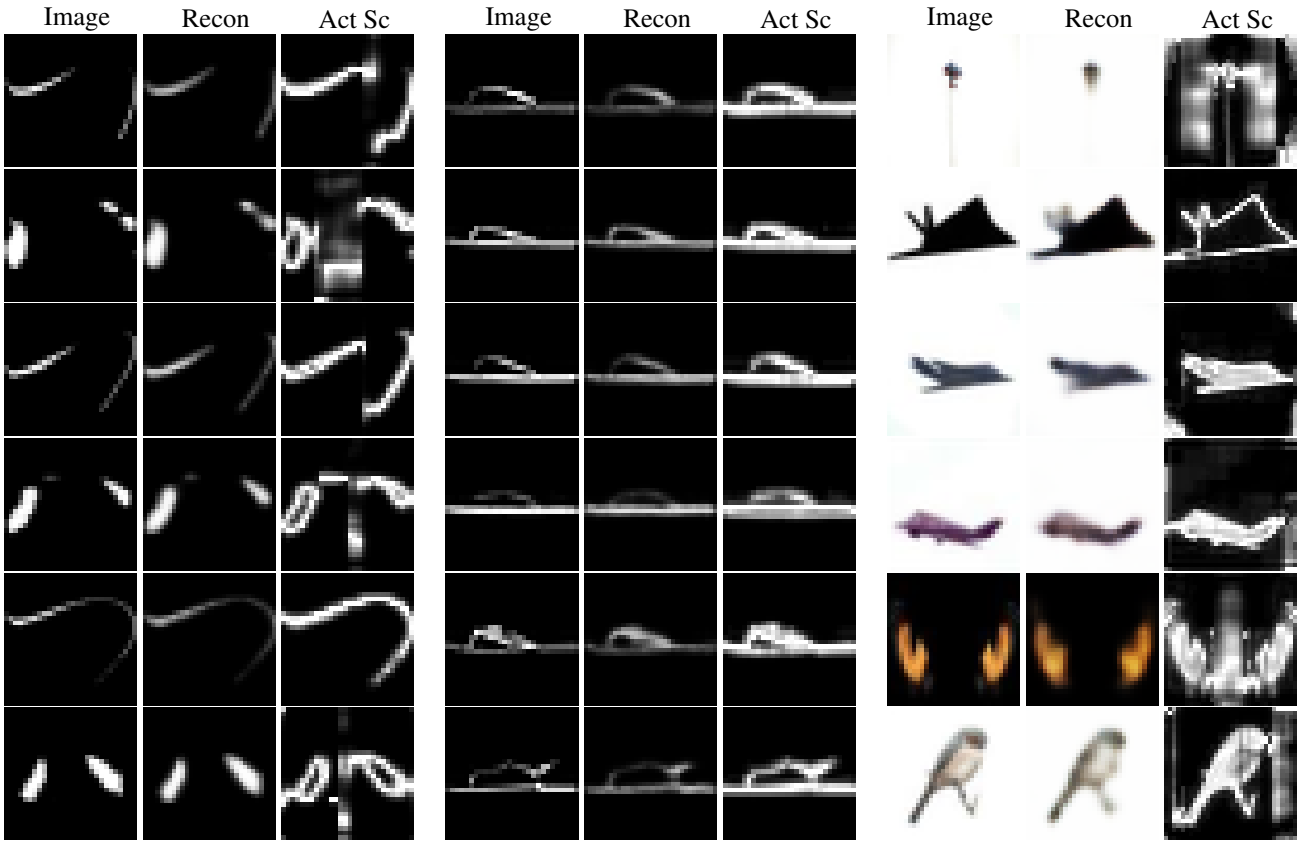


Figure 35. Easiest test samples for KuzushijiMNIST, FashionMNIST and CIFAR10 using our auto-encoder model and our action-score metric. For each dataset we show three columns. The first column from the left corresponds to the real image, the middle one is the reconstruction outputted by our auto-encoder,, and the third one is the per-pixel action score indicating which pixels were the hardest (lighter values) to reconstruct, and which ones were the easiest ones (darker values).

F. Comparison of Action Scores Across Different Models

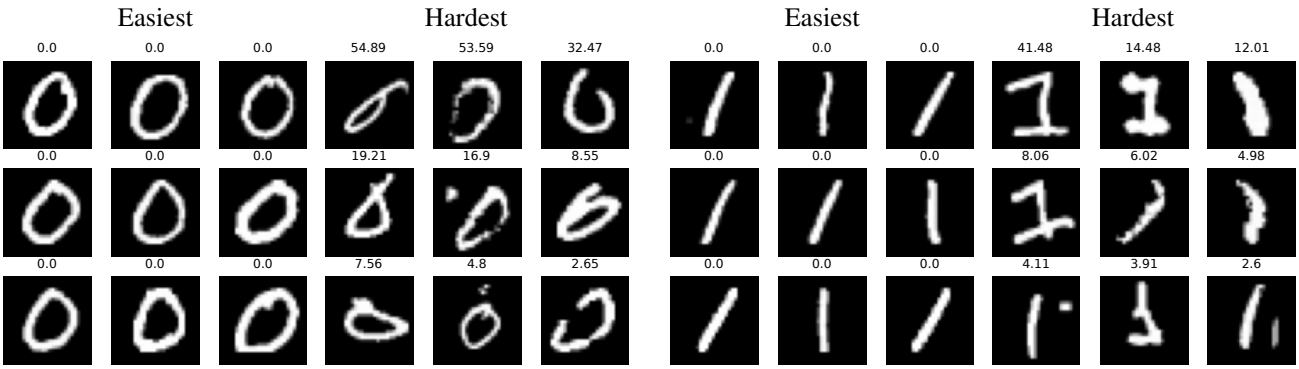
In this section we present additional results that compare action scores for multiple models on the MNIST, CIFAR10, Fashion MNIST, FERPlus, and Kuzushiji MNIST. All comparisons are made in terms to finding the top and bottom nine samples according to their action score produced by a given model, and separated by class, as the action scores for each class might vary in range.

Overall we believe these results show that the action score effectively correlates with difficulty as estimated by a human, hardest samples are usually the most visually complex and hard to classify, while the easiest samples are the simplest and easy to classify.

In all datasets that we evaluated, there are some interesting relationship. For easy samples, they correspond to canonical poses of objects (looking directly at the camera, or common object poses), while the hard samples, there is a large variety of poses and views of each object, which are more unusual, making them harder to classify.

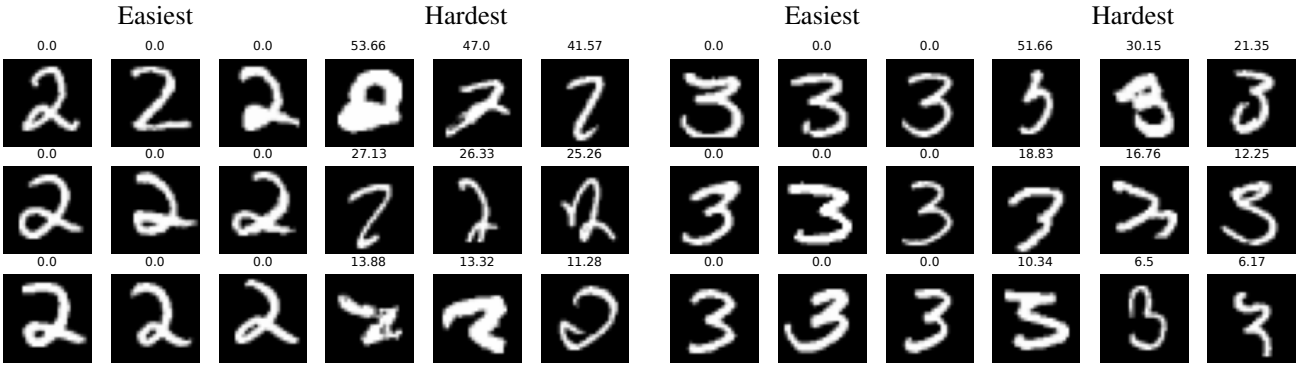
Across models, there is usually some degree of agreement between easy samples, but little agreement on the hardest samples. This is consistent with our findings when comparing model biases.

F.1. MNIST



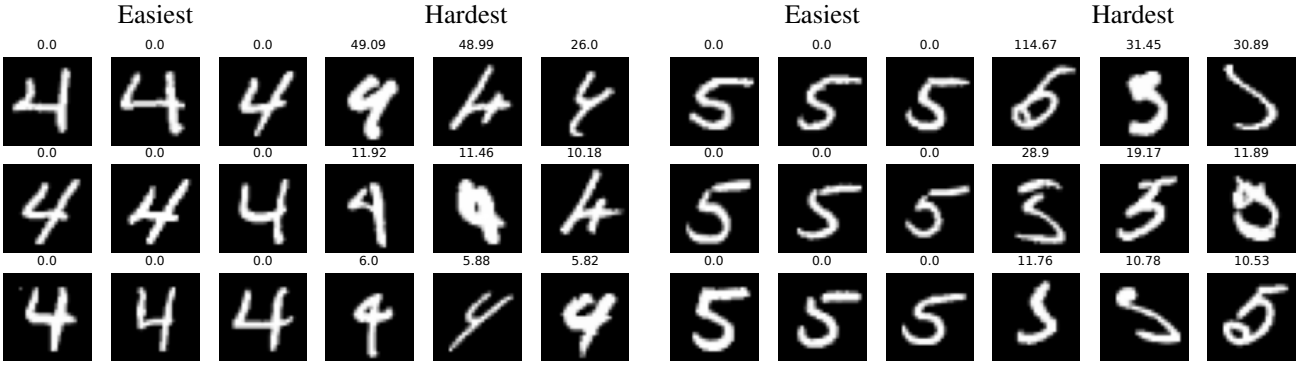
(a) Keras CNN - Class 0 (b) Keras CNN - Class 1

Figure 36. Comparison of easiest (left) and hardest (right) samples on MNIST, Classes 0-1



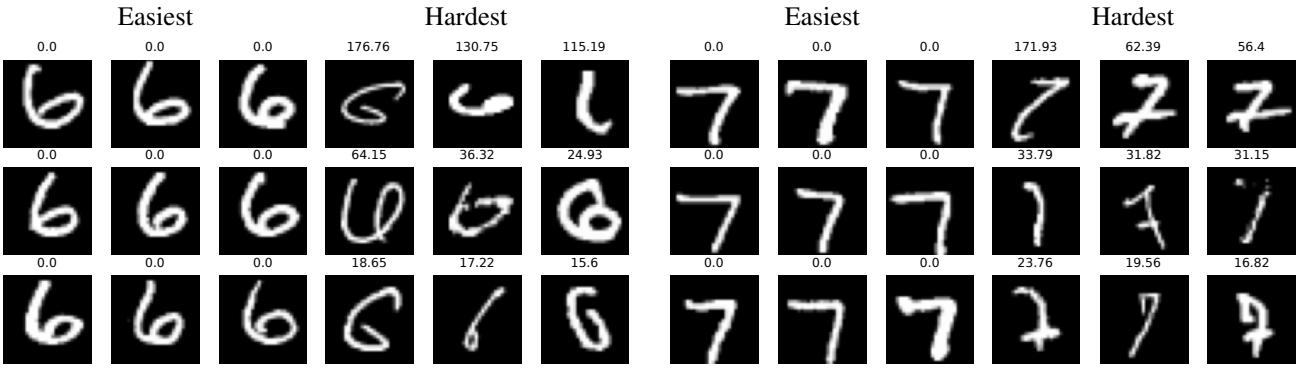
(a) Keras CNN - Class 2 (b) Keras CNN - Class 3

Figure 37. Comparison of easiest (left) and hardest (right) samples on MNIST, Classes 2-3



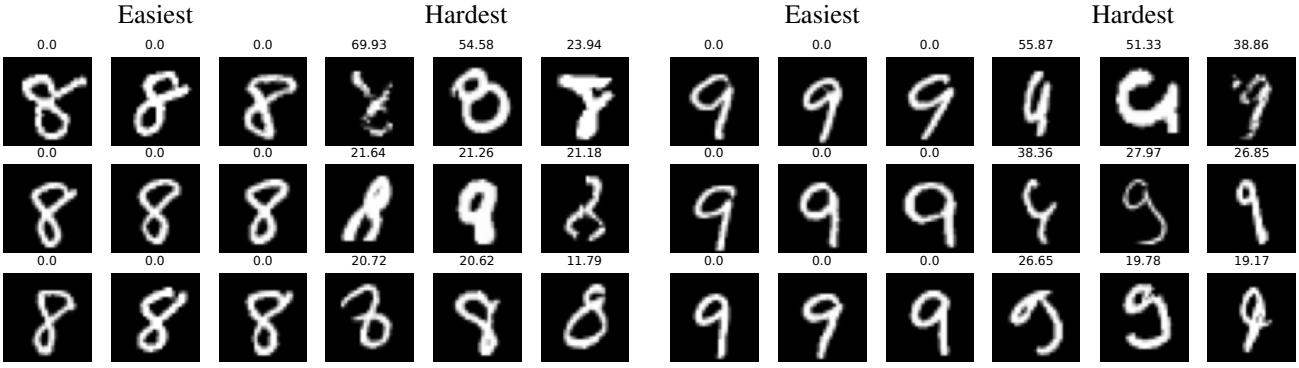
(a) Keras CNN - Class 4 (b) Keras CNN - Class 5

Figure 38. Comparison of easiest (left) and hardest (right) samples on MNIST, Classes 4-5



(a) Keras CNN - Class 6 (b) Keras CNN - Class 7

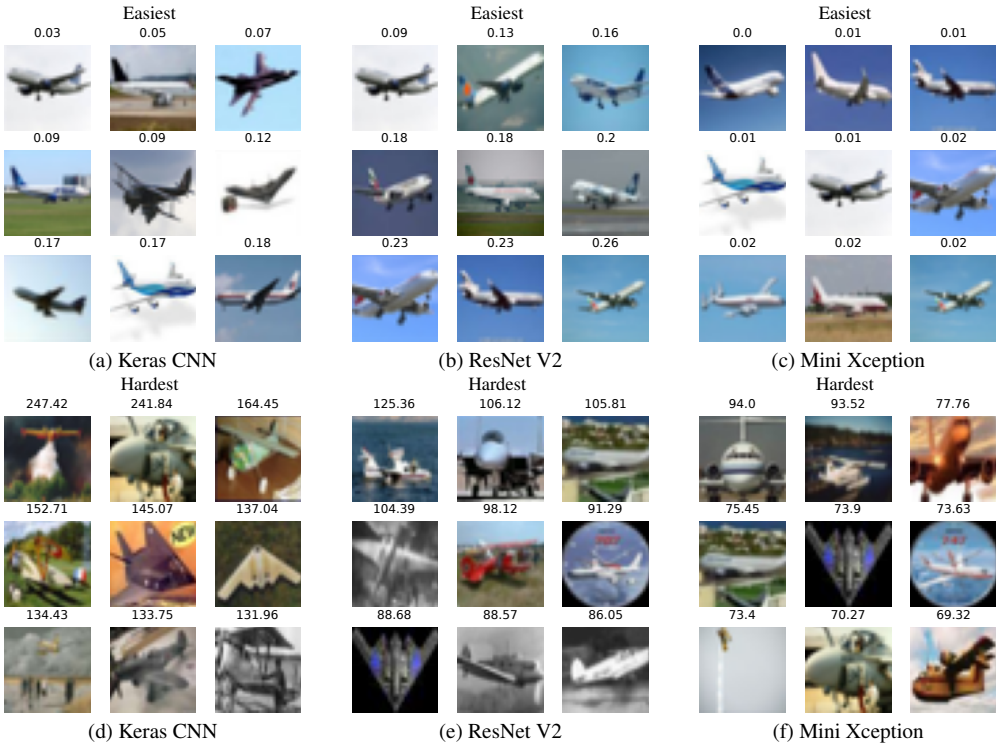
Figure 39. Comparison of easiest (left) and hardest (right) samples on MNIST, Classes 6-7



(a) Keras CNN - Class 8 (b) Keras CNN - Class 9

Figure 40. Comparison of easiest (left) and hardest (right) samples on MNIST, Classes 8-9

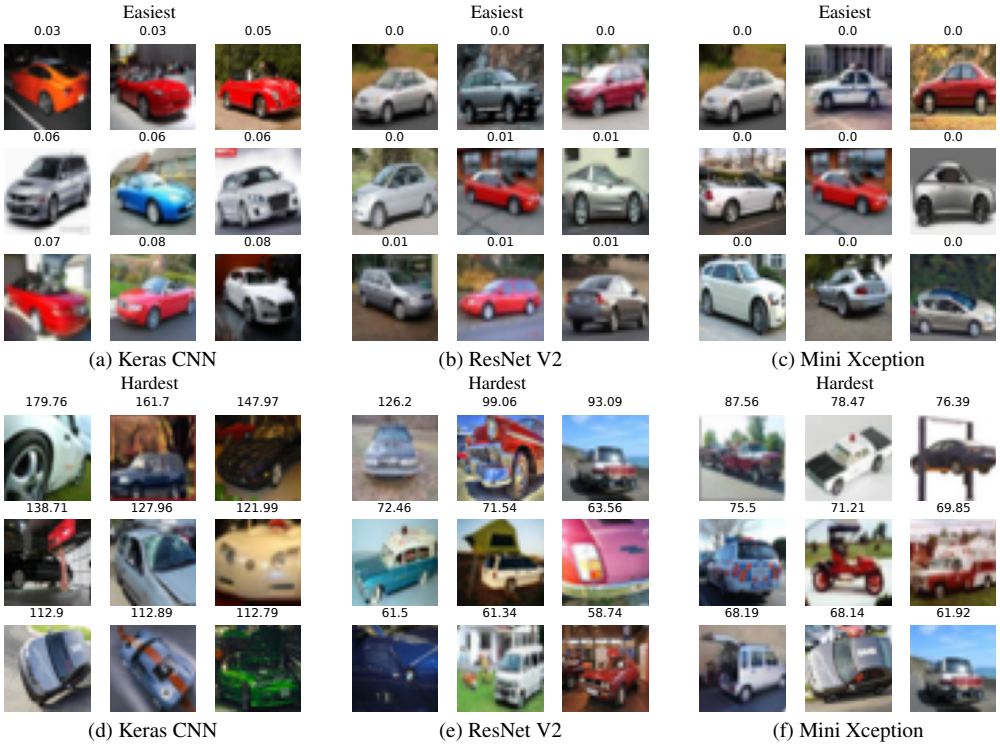
F.2. CIFAR10



(a) Keras CNN (b) ResNet V2 (c) Mini Xception

(d) Keras CNN (e) ResNet V2 (f) Mini Xception

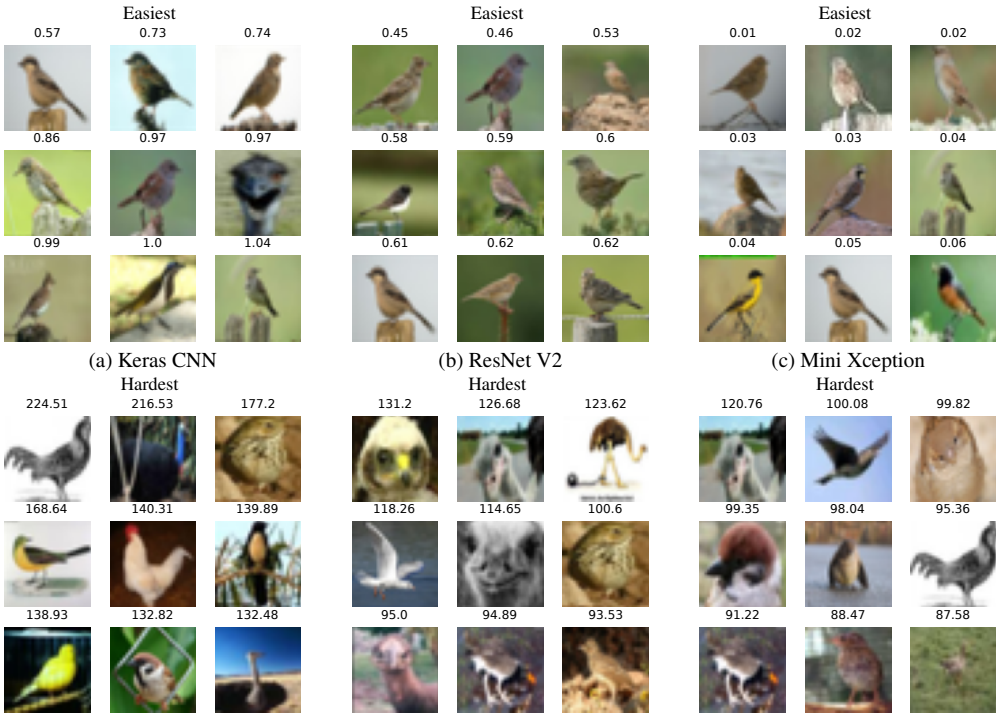
Figure 41. Comparison of easiest (top) and hardest (bottom) samples on CIFAR10, airplane class. On top of each image, their action score is displayed.



(a) Keras CNN (b) ResNet V2 (c) Mini Xception

(d) Keras CNN (e) ResNet V2 (f) Mini Xception

Figure 42. Comparison of easiest (top) and hardest (bottom) samples on CIFAR10, automobile class. On top of each image, their action score is displayed.



(a) Keras CNN (b) ResNet V2 (c) Mini Xception

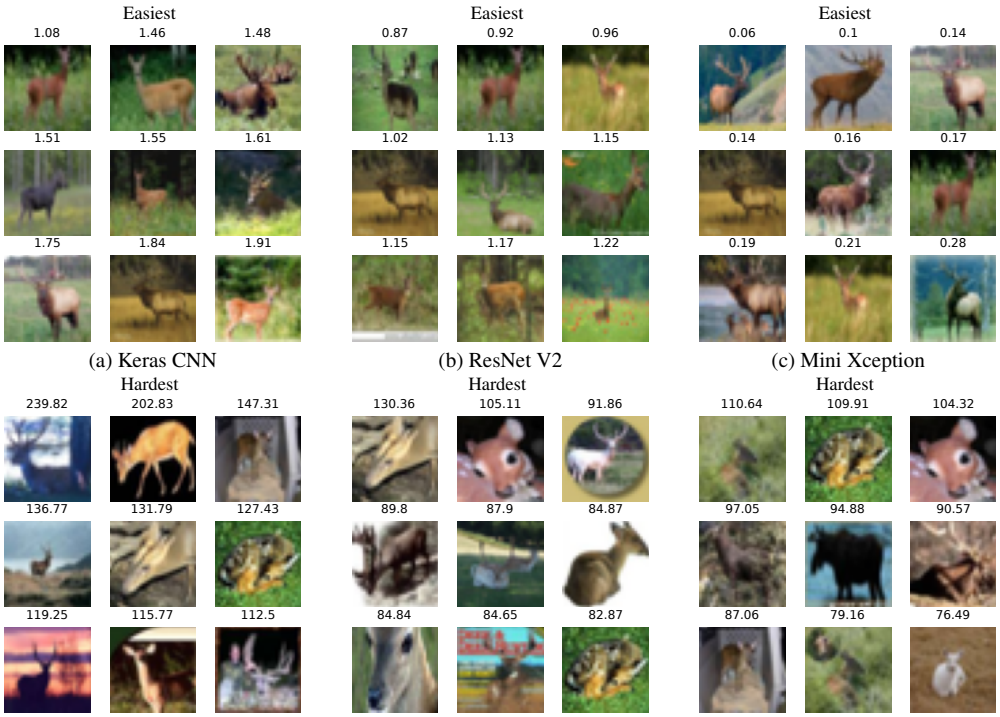
(d) Keras CNN (e) ResNet V2 (f) Mini Xception

Figure 43. Comparison of easiest (top) and hardest (bottom) samples on CIFAR10, bird class. On top of each image, their action score is displayed.

(a) Keras CNN (b) ResNet V2 (c) Mini Xception

(d) Keras CNN (e) ResNet V2 (f) Mini Xception

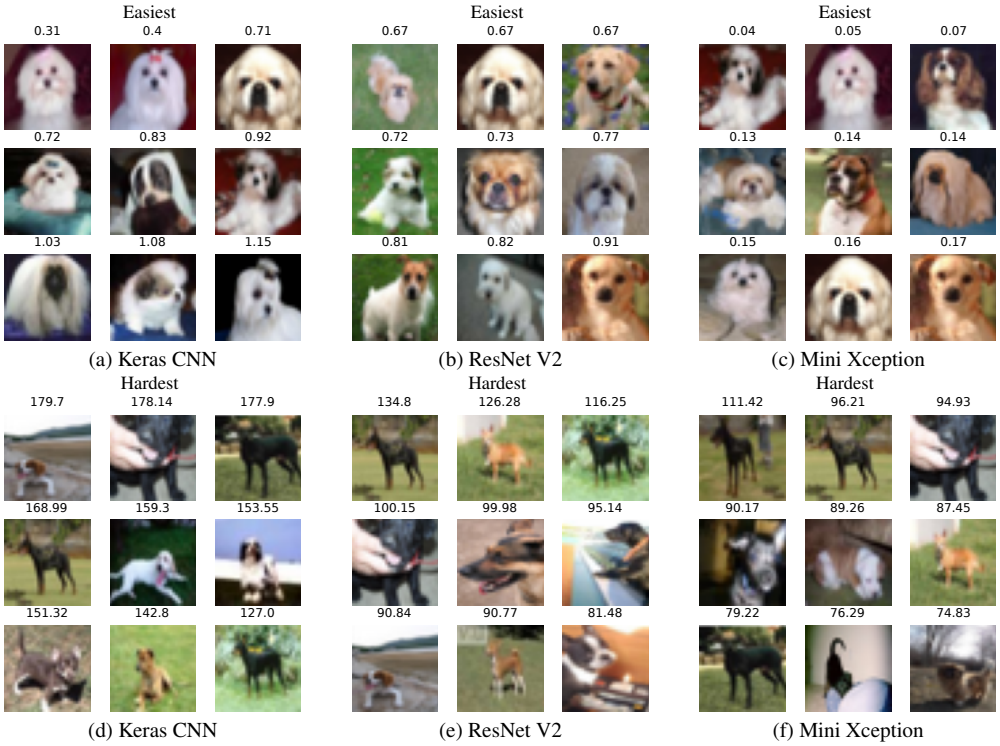
Figure 44. Comparison of easiest (top) and hardest (bottom) samples on CIFAR10, cat class. On top of each image, their action score is displayed.



(a) Keras CNN (b) ResNet V2 (c) Mini Xception

(d) Keras CNN (e) ResNet V2 (f) Mini Xception

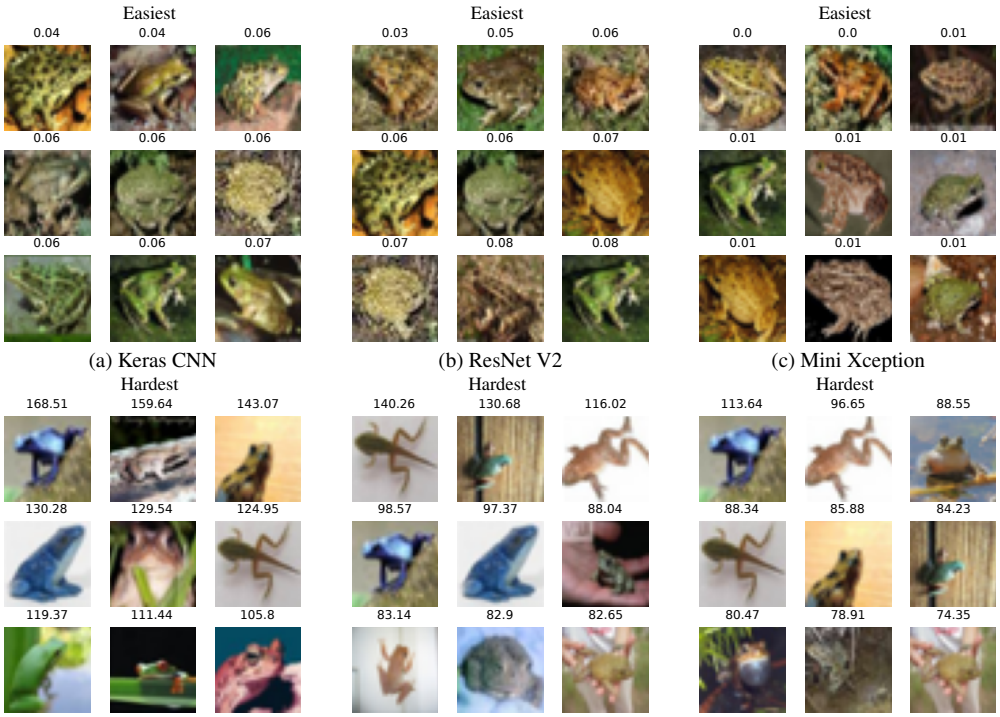
Figure 45. Comparison of easiest (top) and hardest (bottom) samples on CIFAR10, deer class. On top of each image, their action score is displayed.



(a) Keras CNN (b) ResNet V2 (c) Mini Xception

(d) Keras CNN (e) ResNet V2 (f) Mini Xception

Figure 46. Comparison of easiest (top) and hardest (bottom) samples on CIFAR10, dog class. On top of each image, their action score is displayed.



(a) Keras CNN (b) ResNet V2 (c) Mini Xception

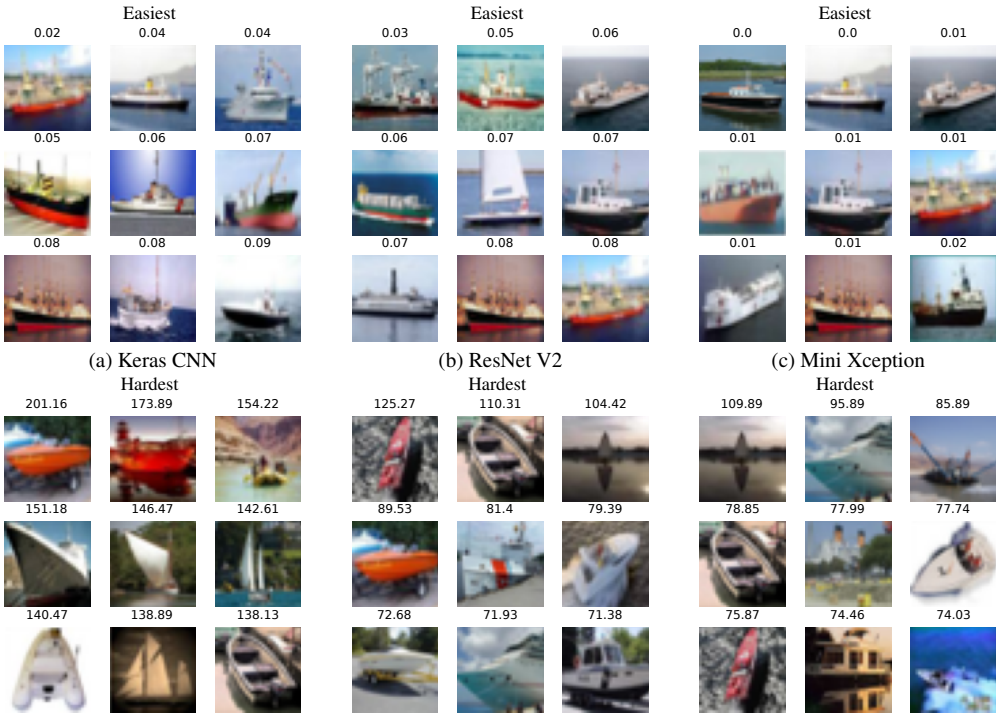
(d) Keras CNN (e) ResNet V2 (f) Mini Xception

Figure 47. Comparison of easiest (top) and hardest (bottom) samples on CIFAR10, frog class. On top of each image, their action score is displayed.

(a) Keras CNN (b) ResNet V2 (c) Mini Xception

(d) Keras CNN (e) ResNet V2 (f) Mini Xception

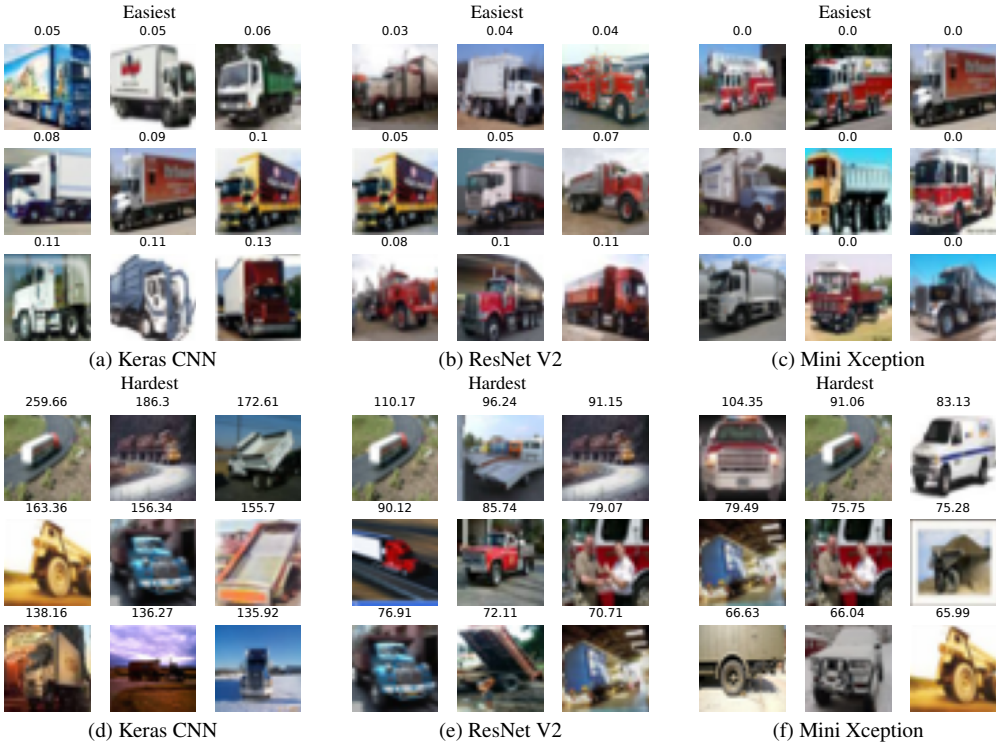
Figure 48. Comparison of easiest (top) and hardest (bottom) samples on CIFAR10, horse class. On top of each image, their action score is displayed.



(a) Keras CNN (b) ResNet V2 (c) Mini Xception

(d) Keras CNN (e) ResNet V2 (f) Mini Xception

Figure 49. Comparison of easiest (top) and hardest (bottom) samples on CIFAR10, ship class. On top of each image, their action score is displayed.

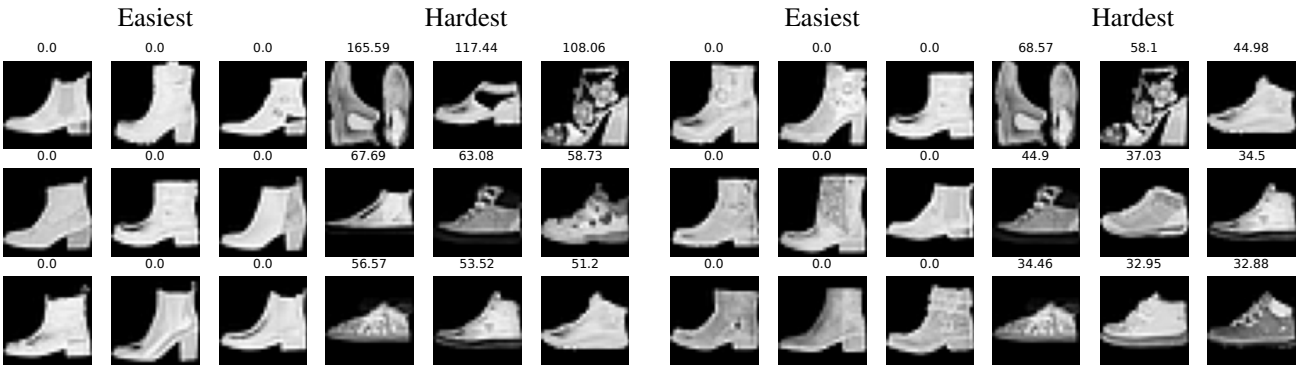


(a) Keras CNN (b) ResNet V2 (c) Mini Xception

(d) Keras CNN (e) ResNet V2 (f) Mini Xception

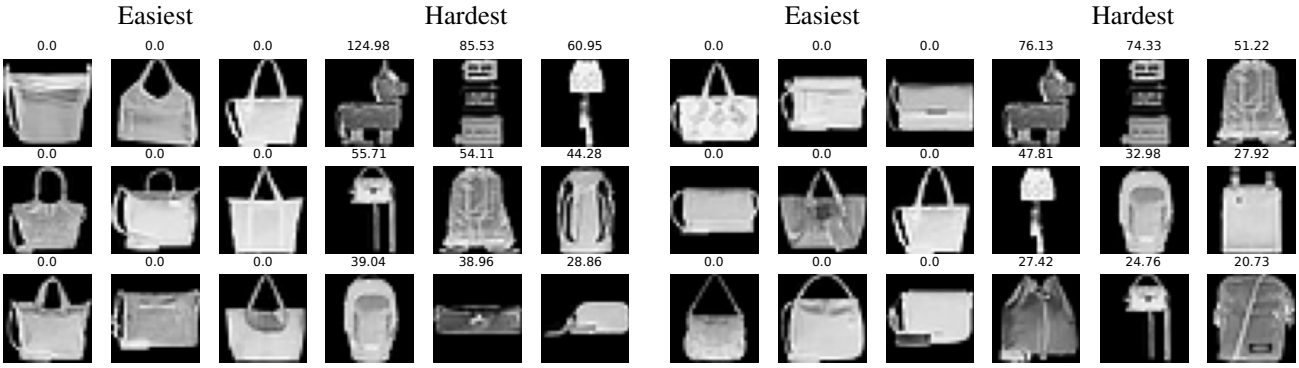
Figure 50. Comparison of easiest (top) and hardest (bottom) samples on CIFAR10, truck class. On top of each image, their action score is displayed.

F.3. Fashion MNIST



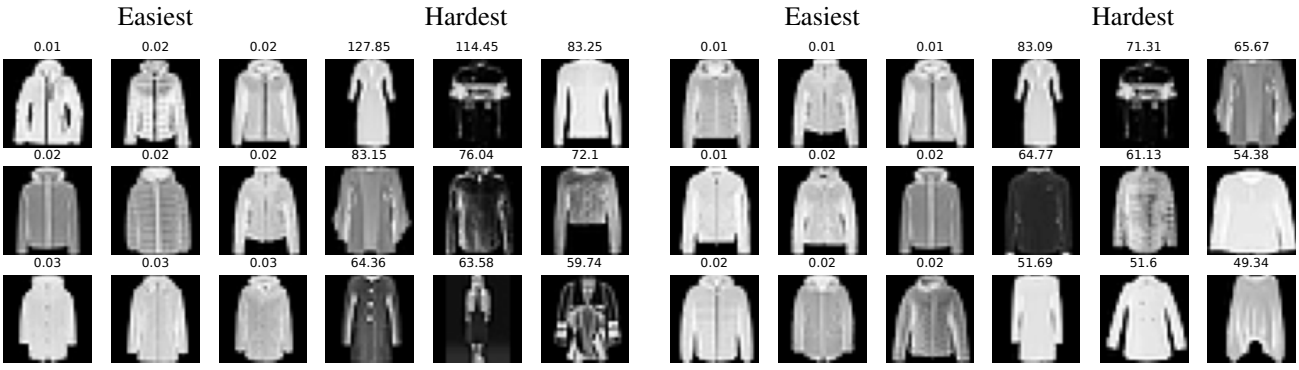
(a) Keras CNN (b) Mini Xception

Figure 51. Comparison of easiest (left) and hardest (right) samples on Fashion MNIST, Ankle boot class



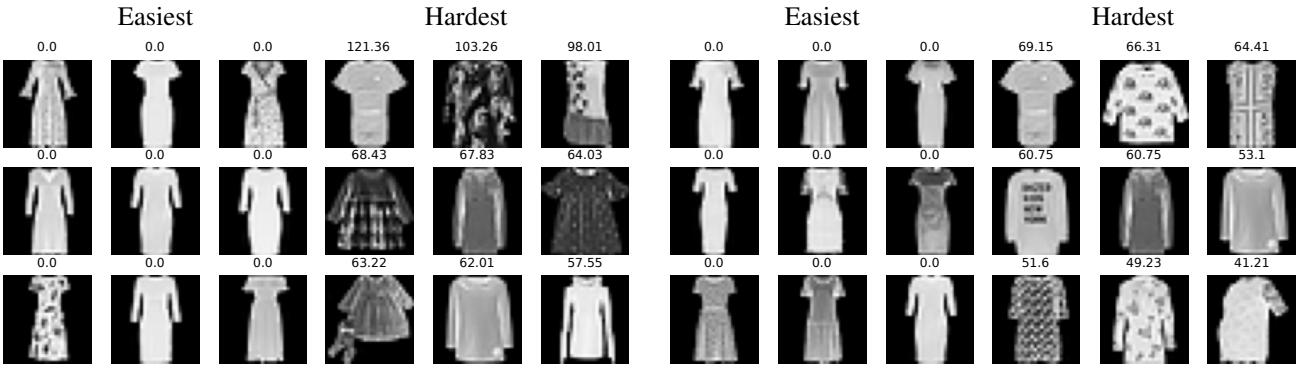
(a) Keras CNN (b) Mini Xception

Figure 52. Comparison of easiest (left) and hardest (right) samples on Fashion MNIST, Bag class



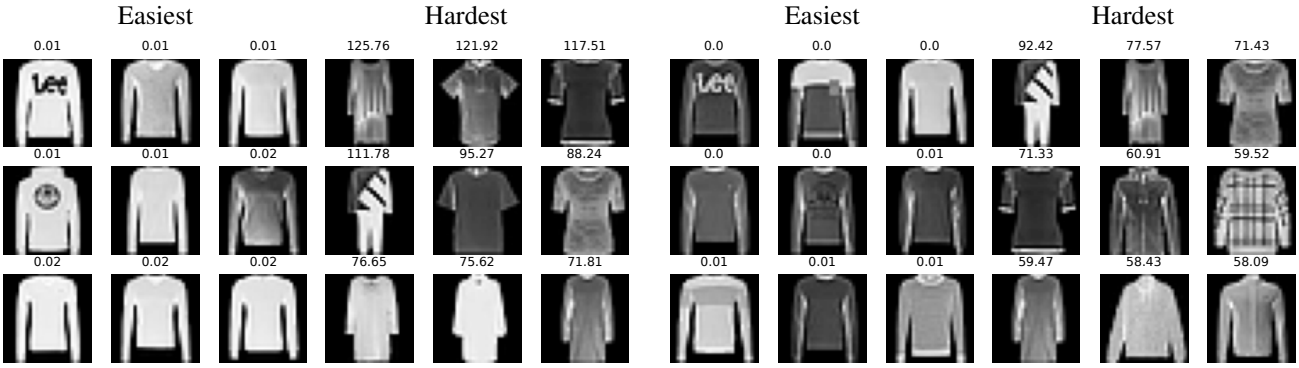
(a) Keras CNN (b) Mini Xception

Figure 53. Comparison of easiest (left) and hardest (right) samples on Fashion MNIST, Coat class



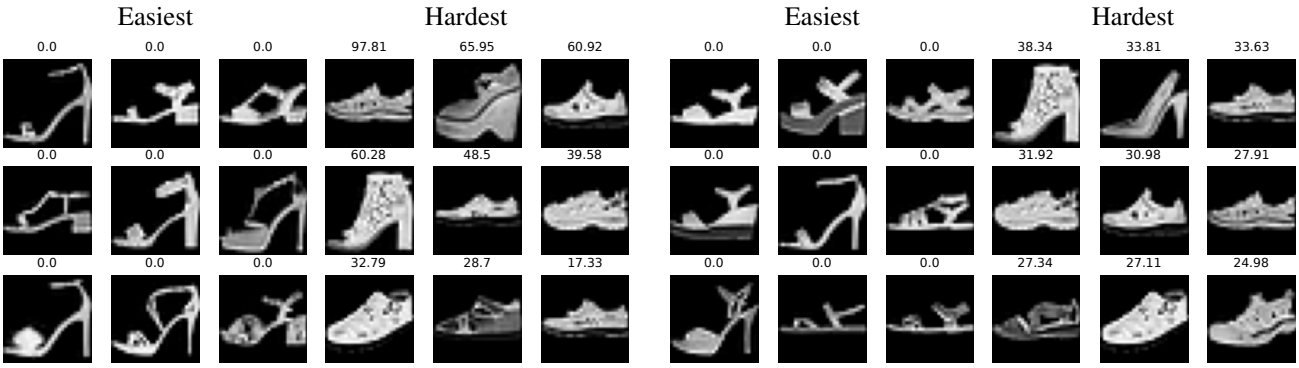
(a) Keras CNN (b) Mini Xception

Figure 54. Comparison of easiest (left) and hardest (right) samples on Fashion MNIST, Dress class



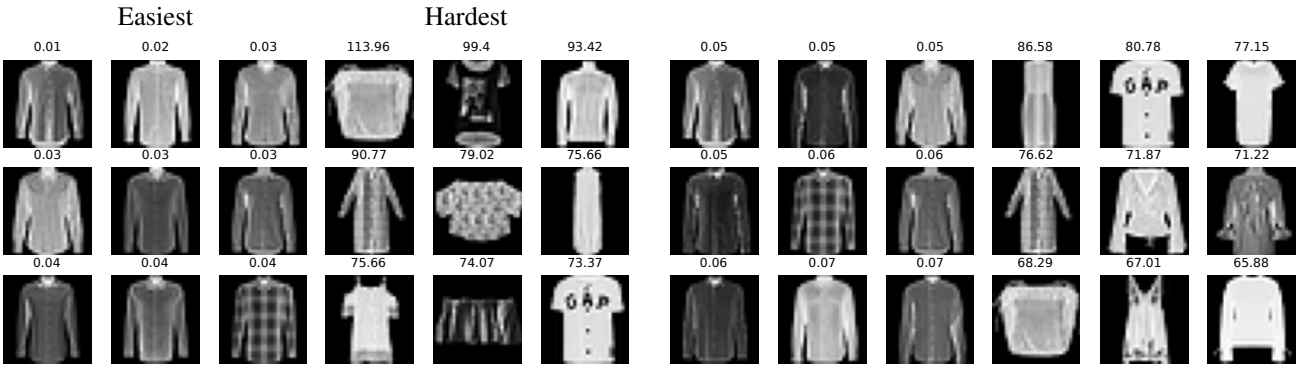
(a) Keras CNN (b) Mini Xception

Figure 55. Comparison of easiest (left) and hardest (right) samples on Fashion MNIST, Pullover class



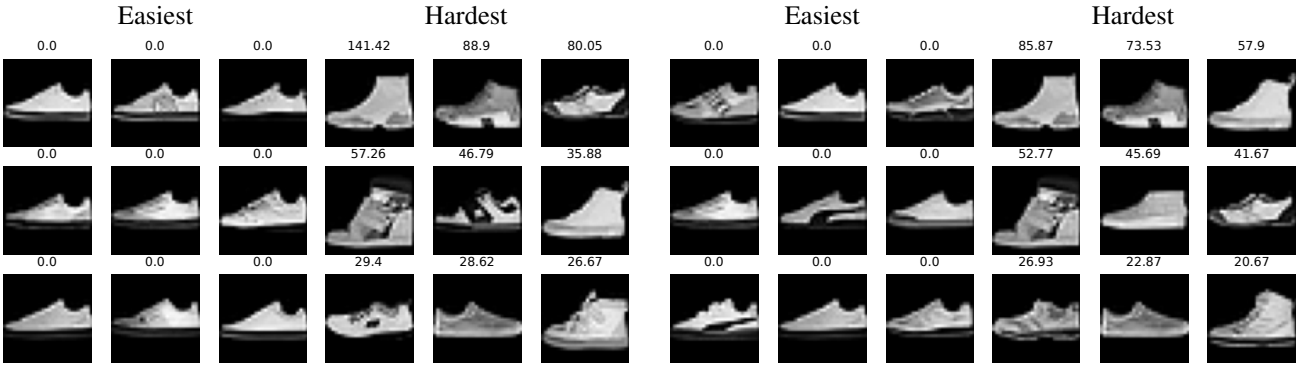
(a) Keras CNN (b) Mini Xception

Figure 56. Comparison of easiest (left) and hardest (right) samples on Fashion MNIST, Sandal class



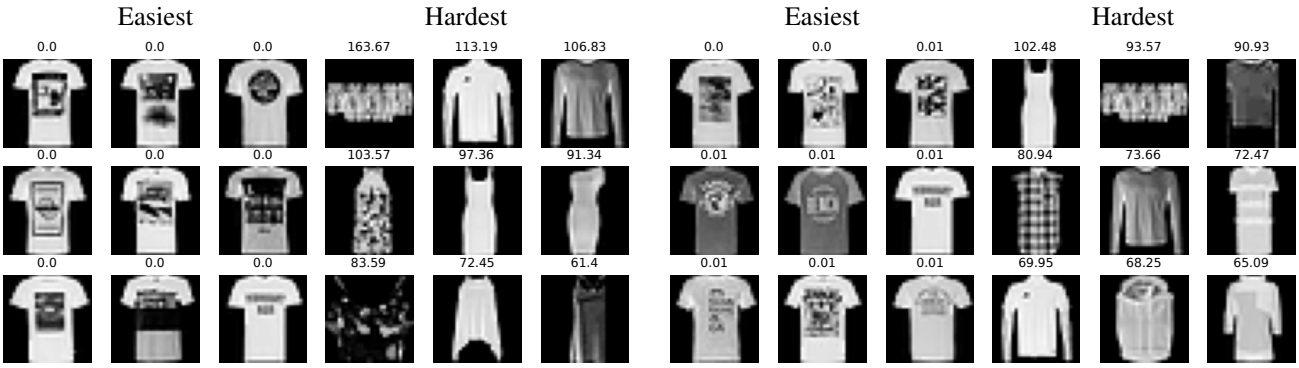
(a) Keras CNN (b) Mini Xception

Figure 57. Comparison of easiest (left) and hardest (right) samples on Fashion MNIST, Shirt class



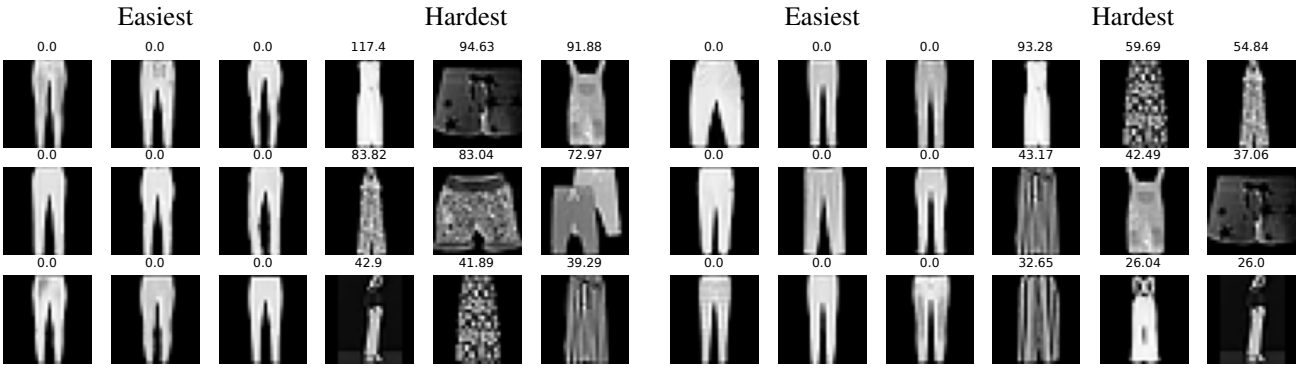
(a) Keras CNN (b) Mini Xception

Figure 58. Comparison of easiest (left) and hardest (right) samples on Fashion MNIST, Sneaker class



(a) Keras CNN (b) Mini Xception

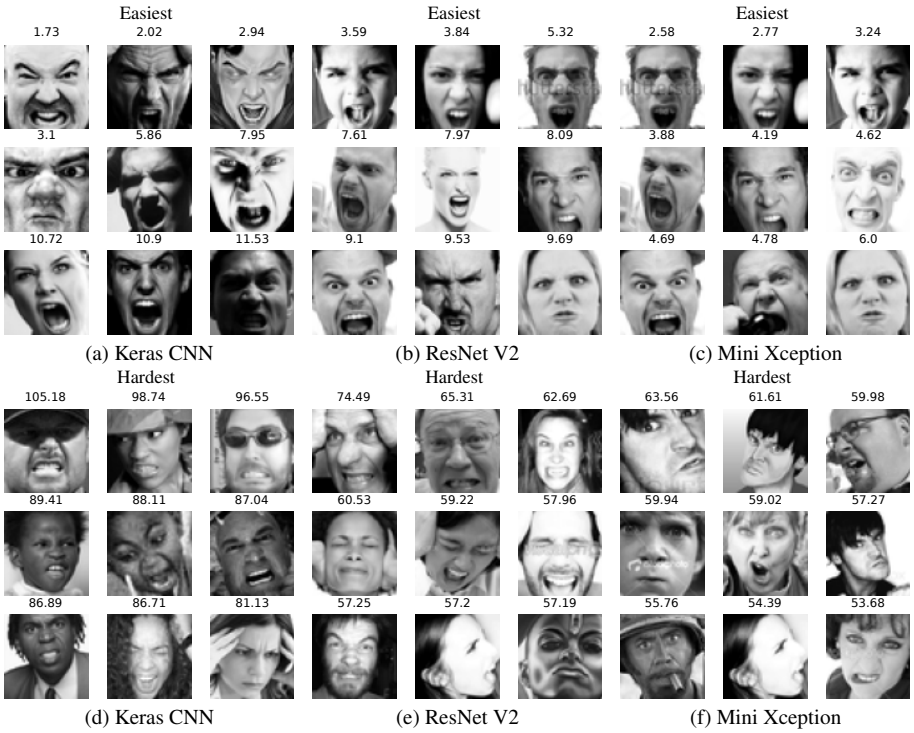
Figure 59. Comparison of easiest (left) and hardest (right) samples on Fashion MNIST, T-shirt/top class



(a) Keras CNN (b) Mini Xception

Figure 60. Comparison of easiest (left) and hardest (right) samples on Fashion MNIST, Trouser class

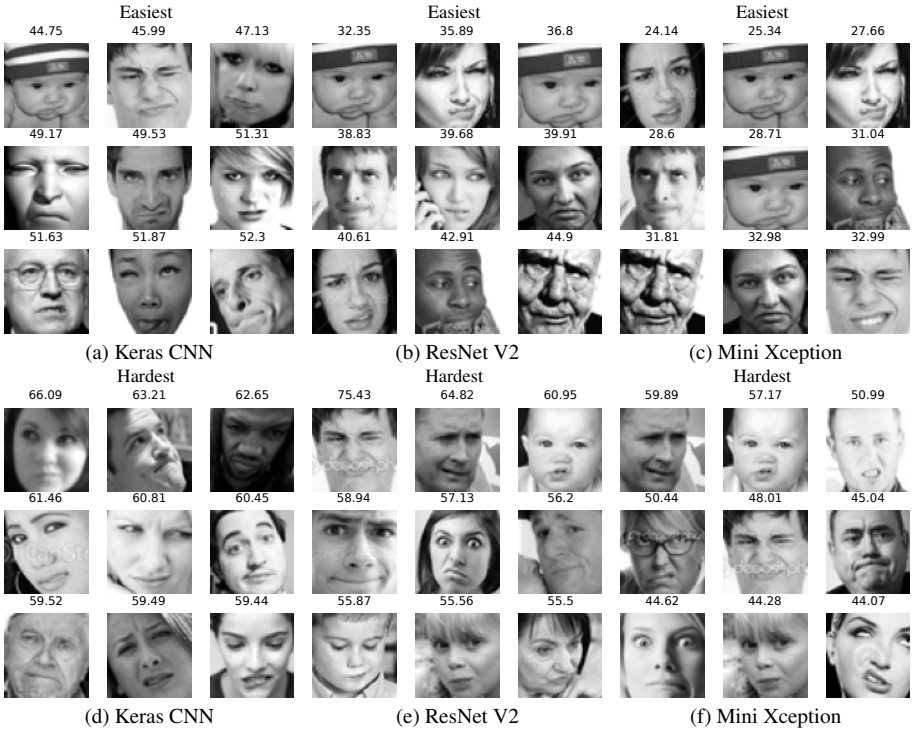
F.4. FERPlus



(a) Keras CNN (b) ResNet V2 (c) Mini Xception

(d) Keras CNN (e) ResNet V2 (f) Mini Xception

Figure 61. Comparison of easiest (top) and hardest (bottom) samples on FERPlus, anger class. On top of each image, their action score is displayed.



(a) Keras CNN (b) ResNet V2 (c) Mini Xception

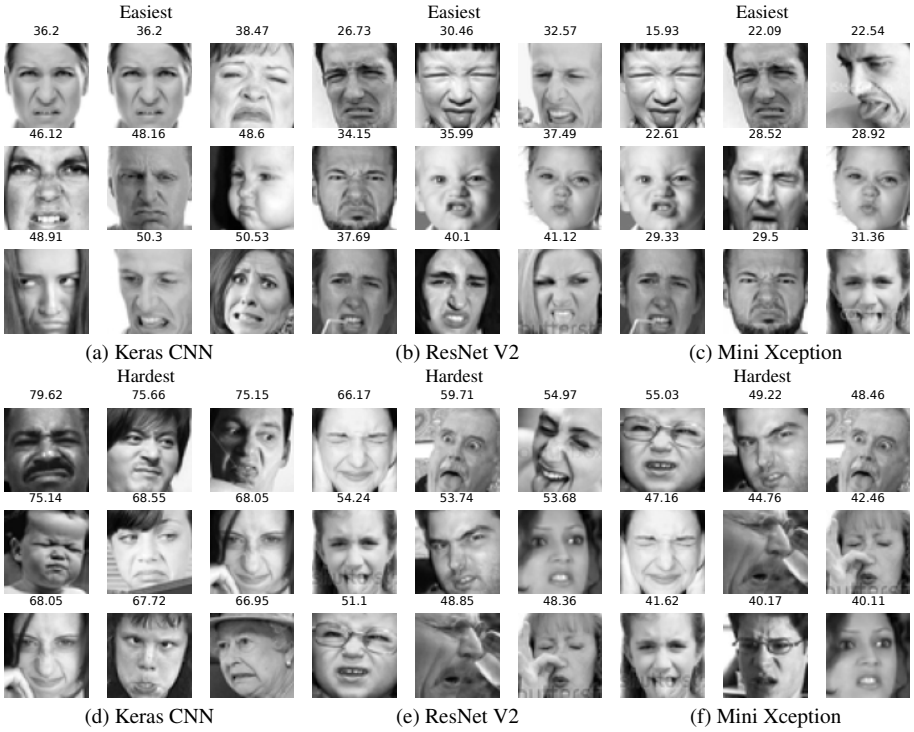
(d) Keras CNN (e) ResNet V2 (f) Mini Xception

Figure 62. Comparison of easiest (top) and hardest (bottom) samples on FERPlus, contempt class. On top of each image, their action score is displayed.

(a) Keras CNN (b) ResNet V2 (c) Mini Xception

(d) Keras CNN (e) ResNet V2 (f) Mini Xception

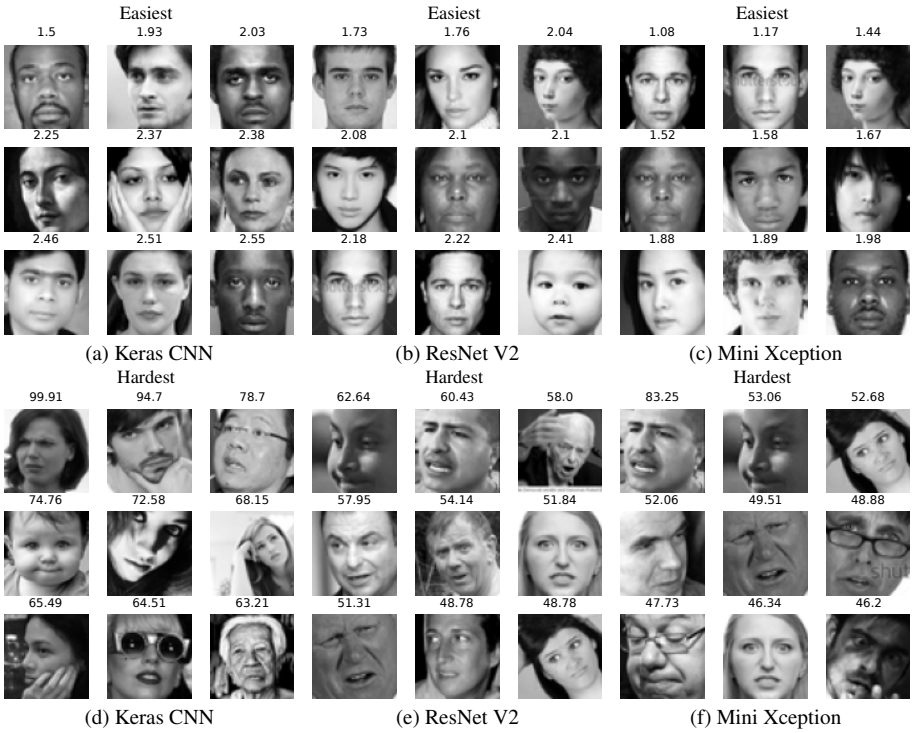
Figure 65. Comparison of easiest (top) and hardest (bottom) samples on FERPlus, happiness class. On top of each image, their action score is displayed.



(a) Keras CNN (b) ResNet V2 (c) Mini Xception

(d) Keras CNN (e) ResNet V2 (f) Mini Xception

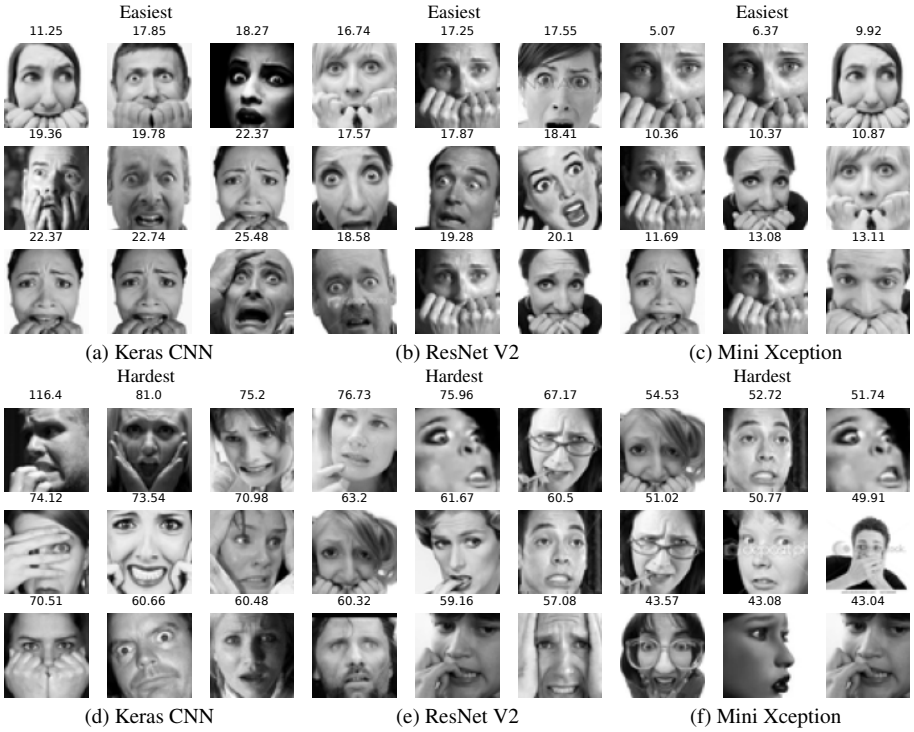
Figure 63. Comparison of easiest (top) and hardest (bottom) samples on FERPlus, disgust class. On top of each image, their action score is displayed.



(a) Keras CNN (b) ResNet V2 (c) Mini Xception

(d) Keras CNN (e) ResNet V2 (f) Mini Xception

Figure 66. Comparison of easiest (top) and hardest (bottom) samples on FERPlus, neutral class. On top of each image, their action score is displayed.



(a) Keras CNN (b) ResNet V2 (c) Mini Xception

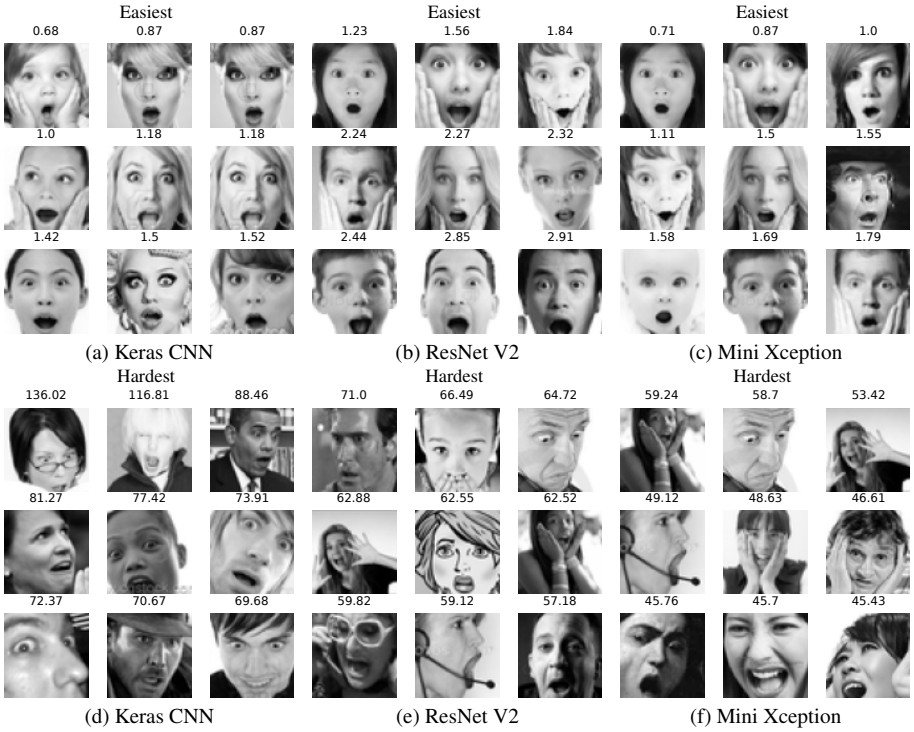
(d) Keras CNN (e) ResNet V2 (f) Mini Xception

Figure 64. Comparison of easiest (top) and hardest (bottom) samples on FERPlus, fear class. On top of each image, their action score is displayed.

(a) Keras CNN (b) ResNet V2 (c) Mini Xception

(d) Keras CNN (e) ResNet V2 (f) Mini Xception

Figure 67. Comparison of easiest (top) and hardest (bottom) samples on FERPlus, sadness class. On top of each image, their action score is displayed.

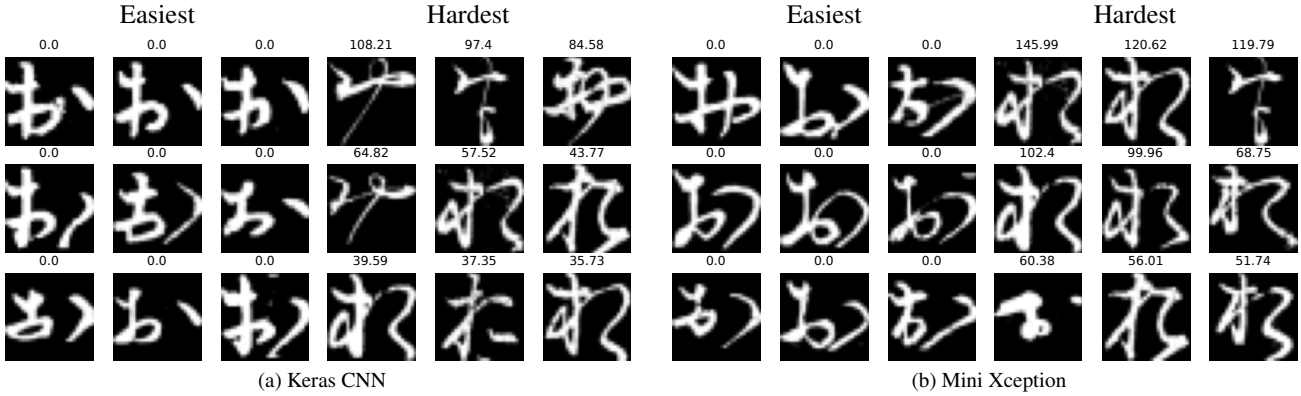


(a) Keras CNN (b) ResNet V2 (c) Mini Xception

(d) Keras CNN (e) ResNet V2 (f) Mini Xception

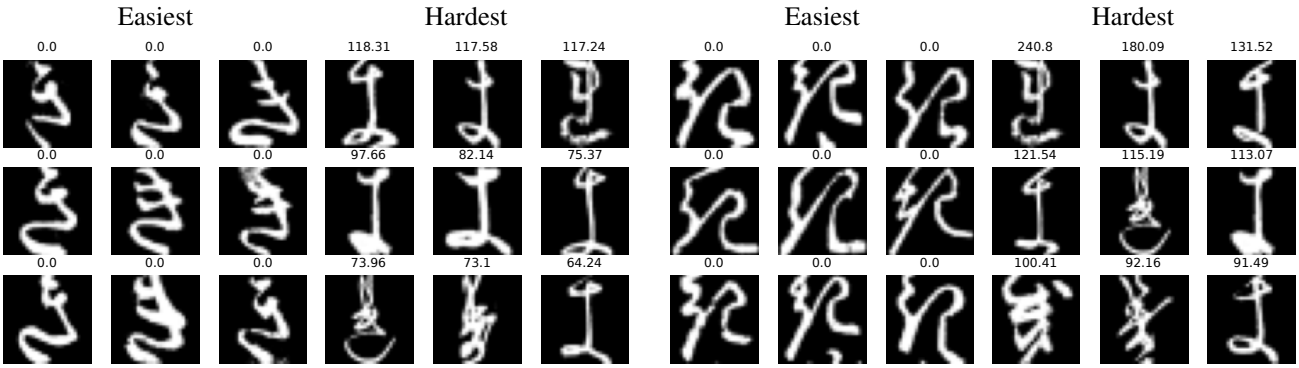
Figure 68. Comparison of easiest (top) and hardest (bottom) samples on FERPlus, surprise class. On top of each image, their action score is displayed.

F.5. Kuzushiji-MNIST



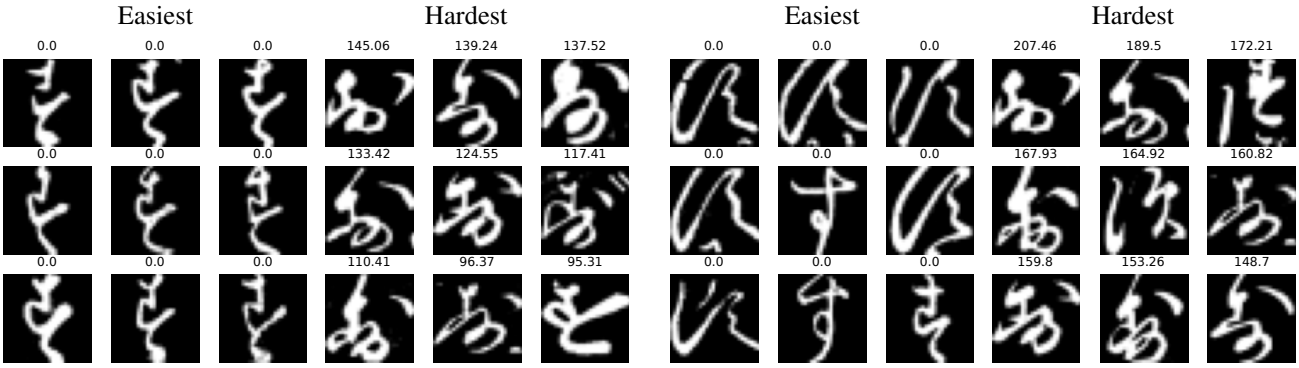
(a) Keras CNN (b) Mini Xception

Figure 69. Comparison of easiest (left) and hardest (right) samples on Kuzushiji-MNIST, お class



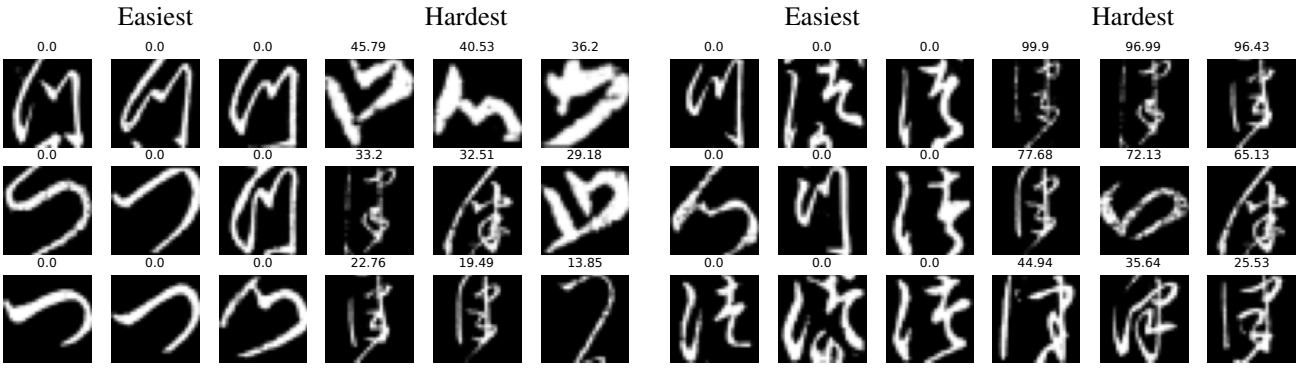
(a) Keras CNN (b) Mini Xception

Figure 70. Comparison of easiest (left) and hardest (right) samples on Kuzushiji-MNIST, き class



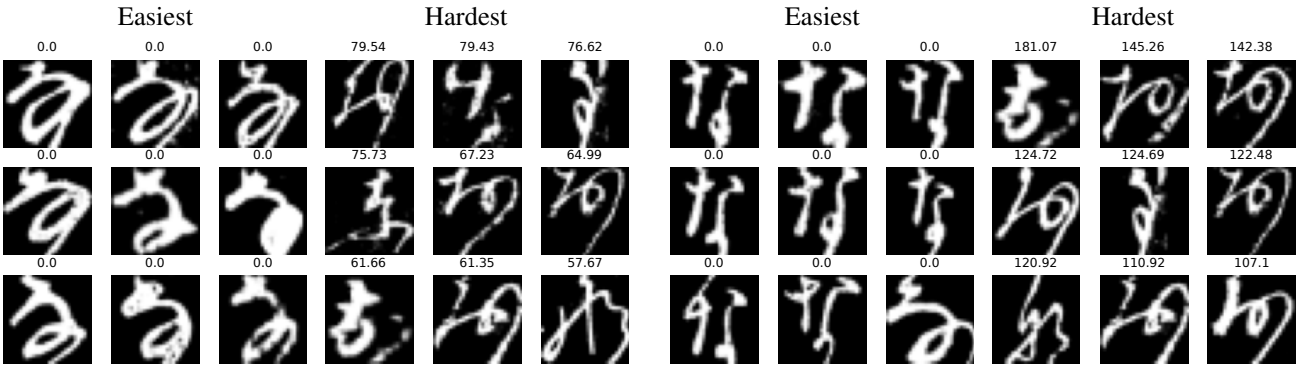
(a) Keras CNN (b) Mini Xception

Figure 71. Comparison of easiest (left) and hardest (right) samples on Kuzushiji-MNIST, す class



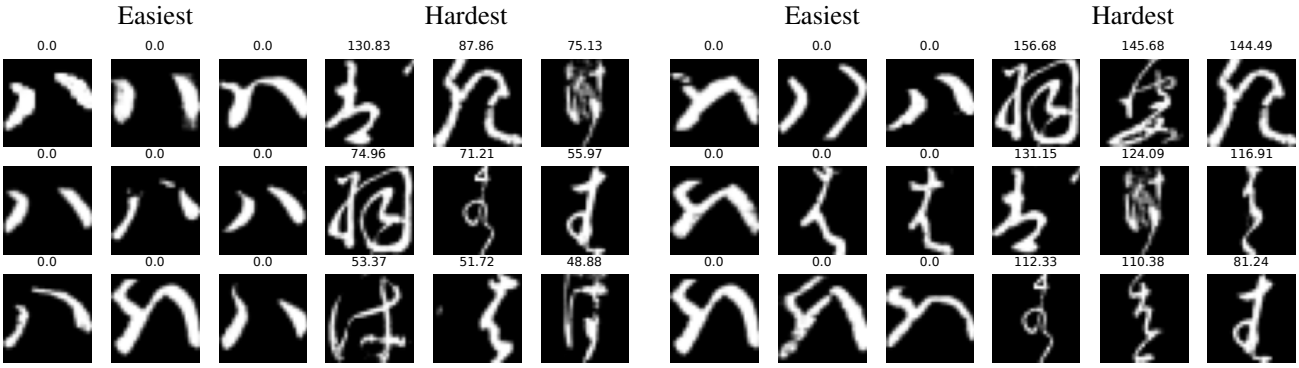
(a) Keras CNN (b) Mini Xception

Figure 72. Comparison of easiest (left) and hardest (right) samples on Kuzushiji-MNIST, つ class



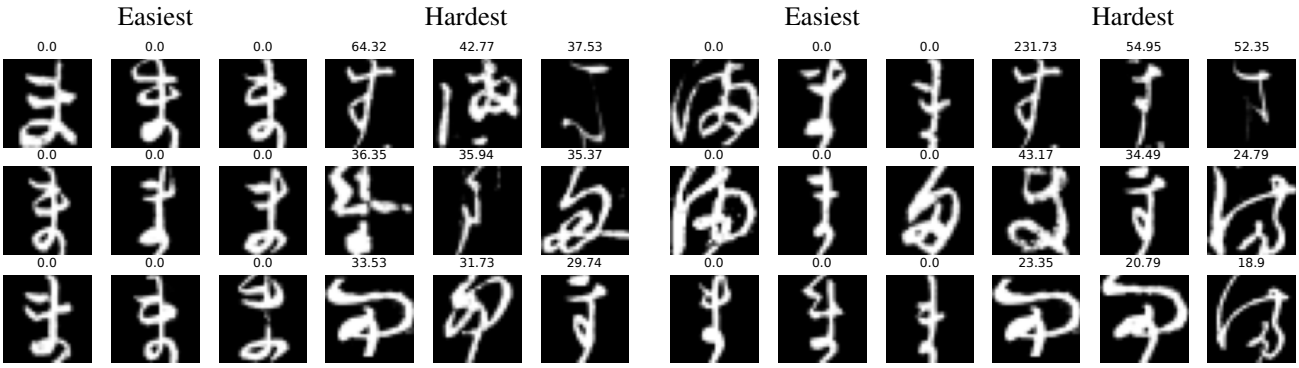
(a) Keras CNN (b) Mini Xception

Figure 73. Comparison of easiest (left) and hardest (right) samples on Kuzushiji-MNIST, な class



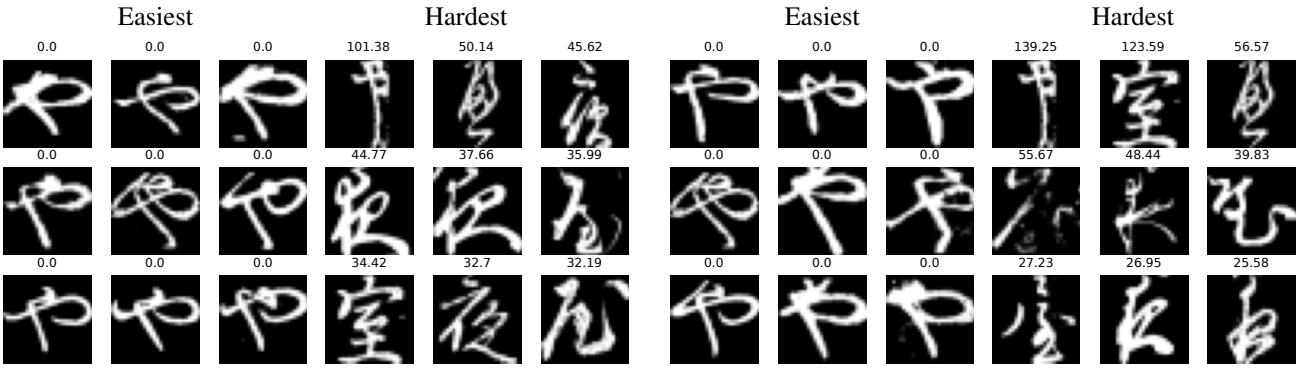
(a) Keras CNN (b) Mini Xception

Figure 74. Comparison of easiest (left) and hardest (right) samples on Kuzushiji-MNIST, は class



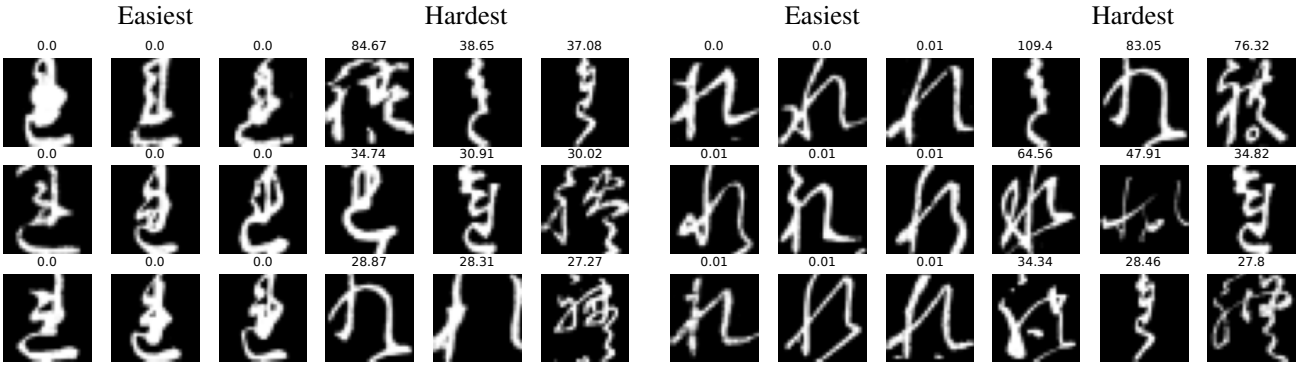
(a) Keras CNN (b) Mini Xception

Figure 75. Comparison of easiest (left) and hardest (right) samples on Kuzushiji-MNIST, ま class



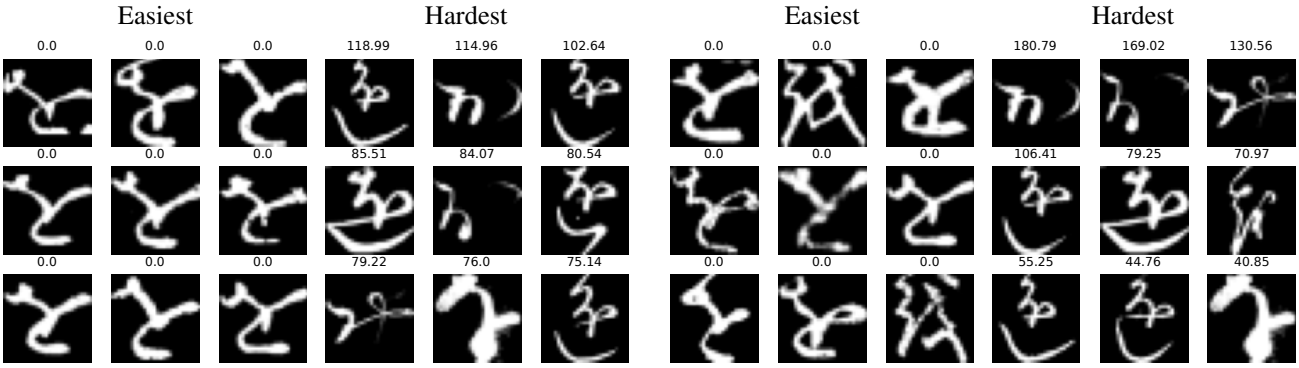
(a) Keras CNN (b) Mini Xception

Figure 76. Comparison of easiest (left) and hardest (right) samples on Kuzushiji-MNIST, や class



(a) Keras CNN (b) Mini Xception

Figure 77. Comparison of easiest (left) and hardest (right) samples on Kuzushiji-MNIST, れ class



(a) Keras CNN (b) Mini Xception

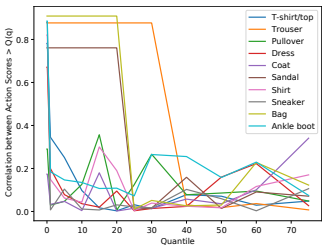
Figure 78. Comparison of easiest (left) and hardest (right) samples on Kuzushiji-MNIST, を class

G. Additional Results for Comparison of Model Biases

In this section we present additional results comparing model biases. These follow the same protocol as the paper (Section 4.6) and compute correlations between action scores produced by each model pair.

G.1. Fashion MNIST

Results for Fashion MNIST are presented in Figure/Table 79b and Figure 79a. Overall these results are consistent with the CIFAR10 results, where the overall and per-class correlation between model's action scores is lower than any correlation with the same model in a second run. The only exception to this pattern seems to be the Bag class. We believe this indicates that the action scores across different models agree less than with their own model (on multiple runs), indicating that each model has its own unique inductive bias, which translates in a different set of easy and hard samples. The per-quantile correlation plot in Figure 79a also indicates that only low action scores are highly correlated, with these correlations tending to zero for higher quantiles.



(a) Keras CNN vs Xception Mini

Models vs Class	Overall	Ankle boot	Bag	Coat	Dress	Pullover	Sandal	Shirt	Sneaker	T-shirt-top	Trouser
CNN-Keras vs Xception-Mini	0.829	0.886	0.909	0.786	0.746	0.877	0.761	0.821	0.880	0.833	0.877
CNN-Keras vs CNN-Keras	0.932	0.922	0.871	0.920	0.916	0.955	0.949	0.920	0.947	0.951	0.932
Xception-Mini vs Xception-Mini	0.898	0.930	0.865	0.888	0.820	0.910	0.768	0.891	0.942	0.917	0.936

(b) Comparison of correlation between action scores for pairs of models, segregated by class, on Fashion MNIST. The bottom rows of this table show baseline correlations between two runs of the same model.

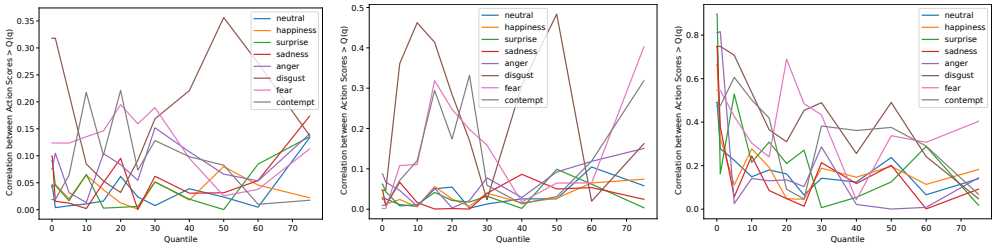
Figure 79. Correlations as a function of the top action scores, where the threshold is a particular quantile of the data. Our results show that correlation is only relatively high for lower action scores, and higher action scores having low correlation. Results on Fashion MNIST.

G.2. FERPlus

Results for FERPlus are presented in Table 3 and Figure 80. These results are interesting as the overall correlation betweeen the Keras-CNN and other models is almost zero, while correlations between the Mini-Xception and ResNetV2 are quite high, but lower than the baselines of the same model over multiple runs. Seems Keras-CNN has a large variance in terms of action scores over multiple runs, indicating that the action scores are completely different for each training run. The overall conclusion for this dataset is the same as CIFAR10 and Fashion MNIST, that action scores over different models only agree for easy examples (low action scores), and disagree for hard examples (high action scores).

Models vs Class	Overall	Anger	Contempt	Disgust	Fear	Happiness	Neutral	Surprise	Sadness
CNN-Keras vs ResnetV2	-0.007	0.089	0.010	0.141	0.057	0.016	0.045	0.080	0.064
CNN-Keras vs Xception-Mini	-0.014	0.115	0.086	0.260	0.003	0.019	0.033	0.041	0.043
Xception-Mini vs ResNetV2	0.909	0.825	0.757	0.775	0.850	0.889	0.922	0.897	0.816
CNN-Keras vs CNN-Keras	-0.023	0.157	0.217	0.019	0.031	0.019	0.006	0.061	0.030
Xception-Mini vs Xception-Mini	0.946	0.903	0.878	0.919	0.874	0.937	0.953	0.953	0.898
ResNetV2 vs ResNetV2	0.951	0.894	0.840	0.928	0.912	0.943	0.947	0.947	0.888

Table 3. Comparison of correlation between action scores for pairs of models, segregated by class, on FERPlus. The bottom rows of this table show baseline correlations between two runs of the same model.



(a) Keras CNN vs ResNet v2 (b) Keras CNN vs Xception Mini (c) ResNet V2 vs Mini Xception

Figure 80. Correlations as a function of the top action scores, where the threshold is a particular quantile of the data. Our results show that correlation is only relatively high for lower action scores, and higher action scores having low correlation. Results on FERPlus.

H. Training Callback Source Code

In this section we present part of our source code, in particular the training callback we use to gather action scores from the training process.

```
import os
import h5py
import numpy as np
from tensorflow.keras.callbacks import Callback
from tensorflow.keras.utils import Progbar

class ScalarActionScore(Callback):
    def __init__(self, sequencer, topic, evaluators, epochs, filepath):
        self.sequencer = sequencer
        self.topic = topic
        self.evaluators = evaluators
        self.epochs = epochs
        self.filepath = filepath

        directory_name = os.path.dirname(self.filepath)
        if not os.path.exists(directory_name):
            os.makedirs(directory_name)

        self.write_file = h5py.File(self.filepath, 'w')
        self.evaluations = self.write_file.create_dataset(
            'evaluations',
            (self.epochs, self.num_samples, self.num_evaluators))

    @property
    def num_evaluators(self):
        return len(self.evaluators)

    @property
    def num_samples(self):
        return len(self.sequencer) * self.sequencer.batch_size

    @property
    def batch_size(self):
        return self.sequencer.batch_size

    def on_epoch_end(self, epoch, logs=None):
        progress_bar = Progbar(len(self.sequencer))
        for batch_index in range(len(self.sequencer)):
            inputs, labels = self.sequencer.__getitem__(batch_index)
            for eval_arg, evaluator in enumerate(self.evaluators):
                batch_arg_A = self.batch_size * (batch_index)
                batch_arg_B = self.batch_size * (batch_index + 1)
                y_true = labels[self.topic]
                y_pred = self.model(inputs)
                evaluation = evaluator(y_true, y_pred)
                self.evaluations[
                    epoch, batch_arg_A:batch_arg_B, eval_arg] = evaluation
            progress_bar.update(batch_index + 1)
        self.evaluations.flush()

    def on_train_end(self, logs=None):
        self.write_file.close()
```