The Pokédex Project

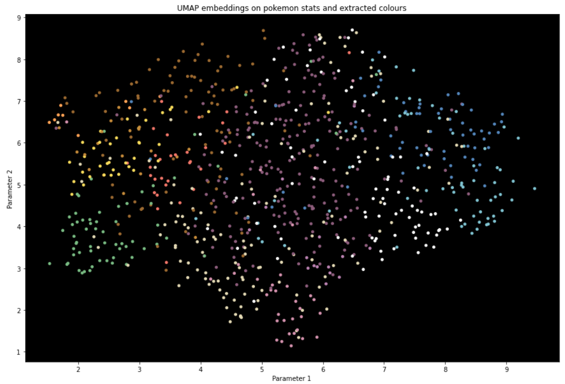
Frederik Cornelius Østergaard, Frederik Ørsted Kjeldal, Tobias Mølgaard Nielsen, Marcus Nørgaard Weng

KØBENHAVNS UNIVERSITET

Overview of the project

Webscraping + Data

- Extracting images from online Pokédex

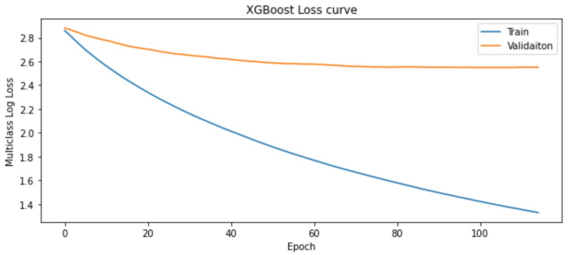


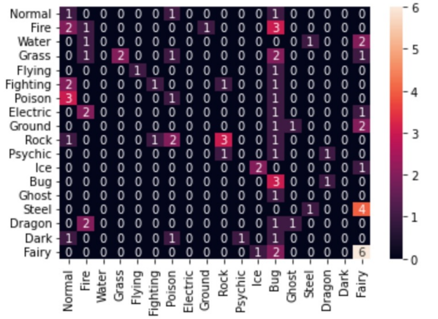
Clustering pokémon images:

- UMAP

Predicting the types of pokémon:

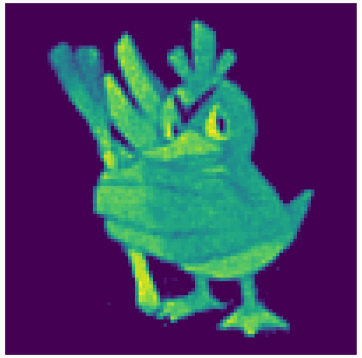
- XGBoost, CNN, Human models





Reconstructing pokémon images:

- Variational Autoencoder



Problem statement

- A new generation of Pokémon (gen 8) is coming soon
- We have not seen them before
- Is is possible to predict the type of the new pokémon?

Generation 1 - 7

787x

…

Generation 8

82x

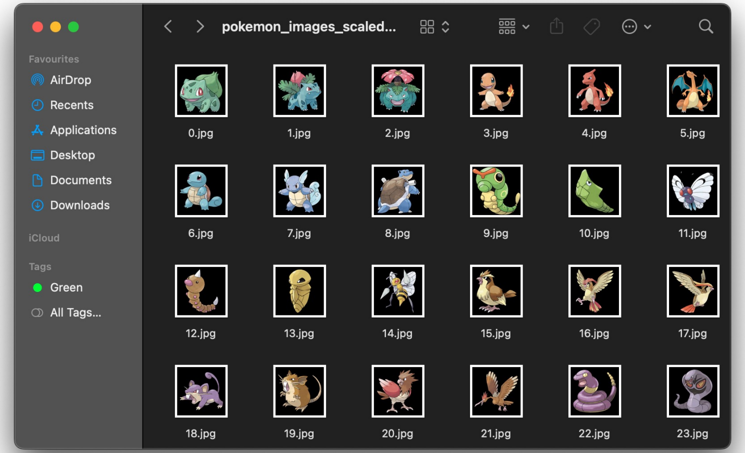
…

Data

Webscraping:

- Download image of each Pokémon from Bulbapedia
- Rescale to 300x300
- Save greyscale copies
- Save stats, shape and abilities of each Pokémon

ID	Name	Type1	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
1	Bulbasaur	Grass	318	45	49	49	65	65	45	1
2	Ivysaur	Grass	405	60	62	63	80	80	60	1
3	Venusaur	Grass	525	80	82	83	100	100	80	1
4	Charmander	Fire	309	39	52	43	60	50	65	1
5	Charmeleon	Fire	405	58	64	58	80	65	80	1
...	...	...	...	...	...	...	...	...	...	...
894	Regieleki	Electric	580	80	100	50	100	50	200	8
895	Regidrago	Dragon	580	200	100	50	100	50	80	8
896	Glastrier	Ice	580	100	145	130	65	110	30	8
897	Spectrier	Ghost	580	100	65	60	145	80	130	8
898	Calyrex	Psychic	500	100	80	80	80	80	80	8



Data

- 869 datapoints

Each datapoint input:

- Image
  - 300x300x3 RGB
  - 270.000 numbers (0-255)
- Data
  - Stats
  - Body shape (1-14)
  - Abilities

Distribution of types

ID	Name	Type1	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Shapes	Ability1	Ability2
1	Bulbasaur	Grass	318	45	49	49	65	65	45	1	8	overgrow	chlorophyll

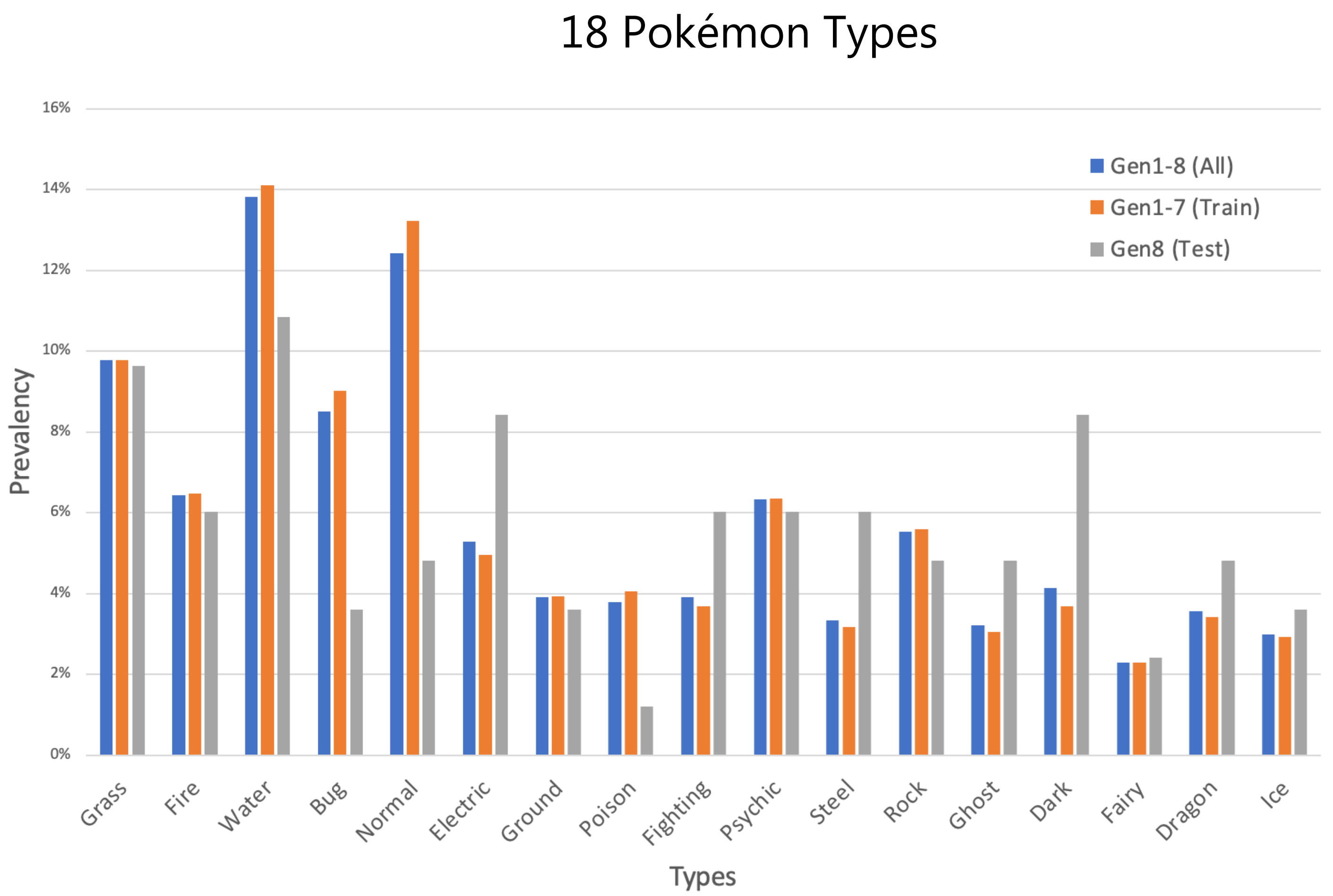
Data

- 869 datapoints

Each datapoint input:

- Image
  - 300x300x3 RGB
  - 270.000 numbers (0-255)
- Data
  - Stats
  - Body shape (1-14)
  - Abilities

Distribution of types



CLUSTERING

Clustering

Model:

UMAP

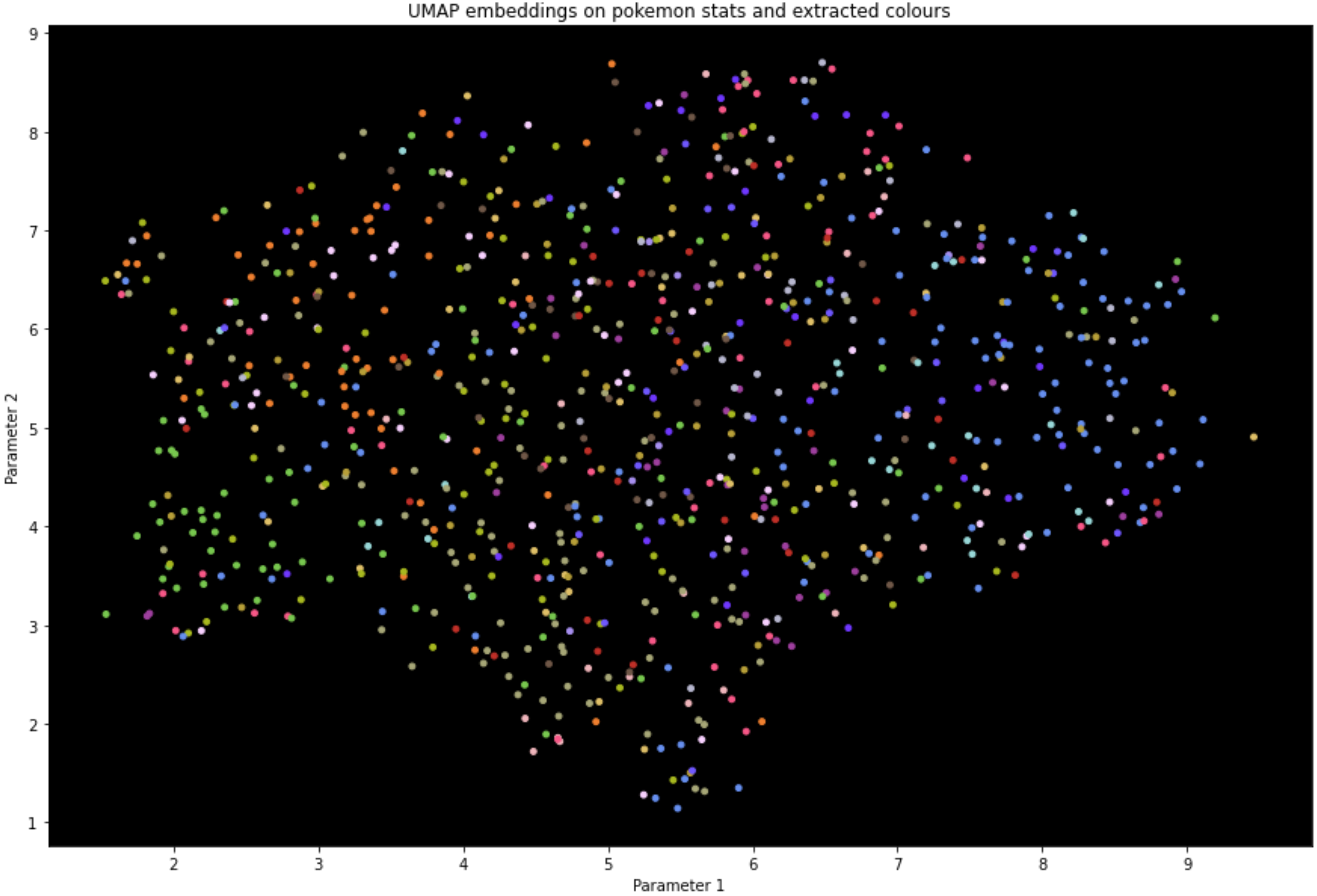
Preprocessing:

- Extracted pixel colours and removed all counts of 'black'
- Encoded Pokémon abilities 1-3

Input:

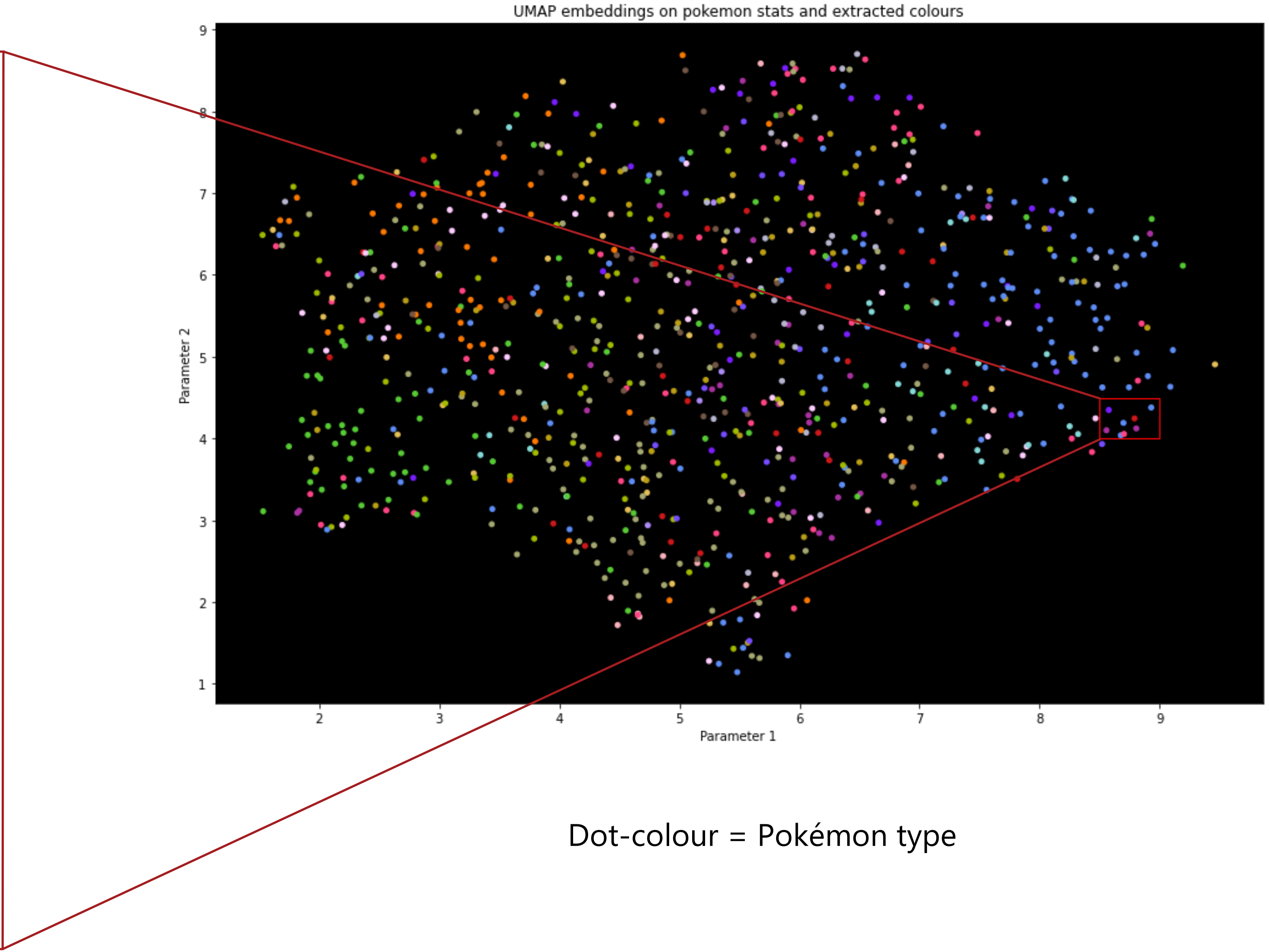
- Number of pixels of each colour
- Shapes
- Stats
- Encoded abilities

HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Shapes	red	...	dark_brown	white	black	gray	pink	light_purple	dark_purple	Ability1	Ability2	Ability3
45	49	49	65	65	45	1	8	2	...	6	12	434	5	3	4	74	95	187	455



Dot-colour = Pokemon type

Clustering



Dot-colour = Pokémon type

Clustering

Model:

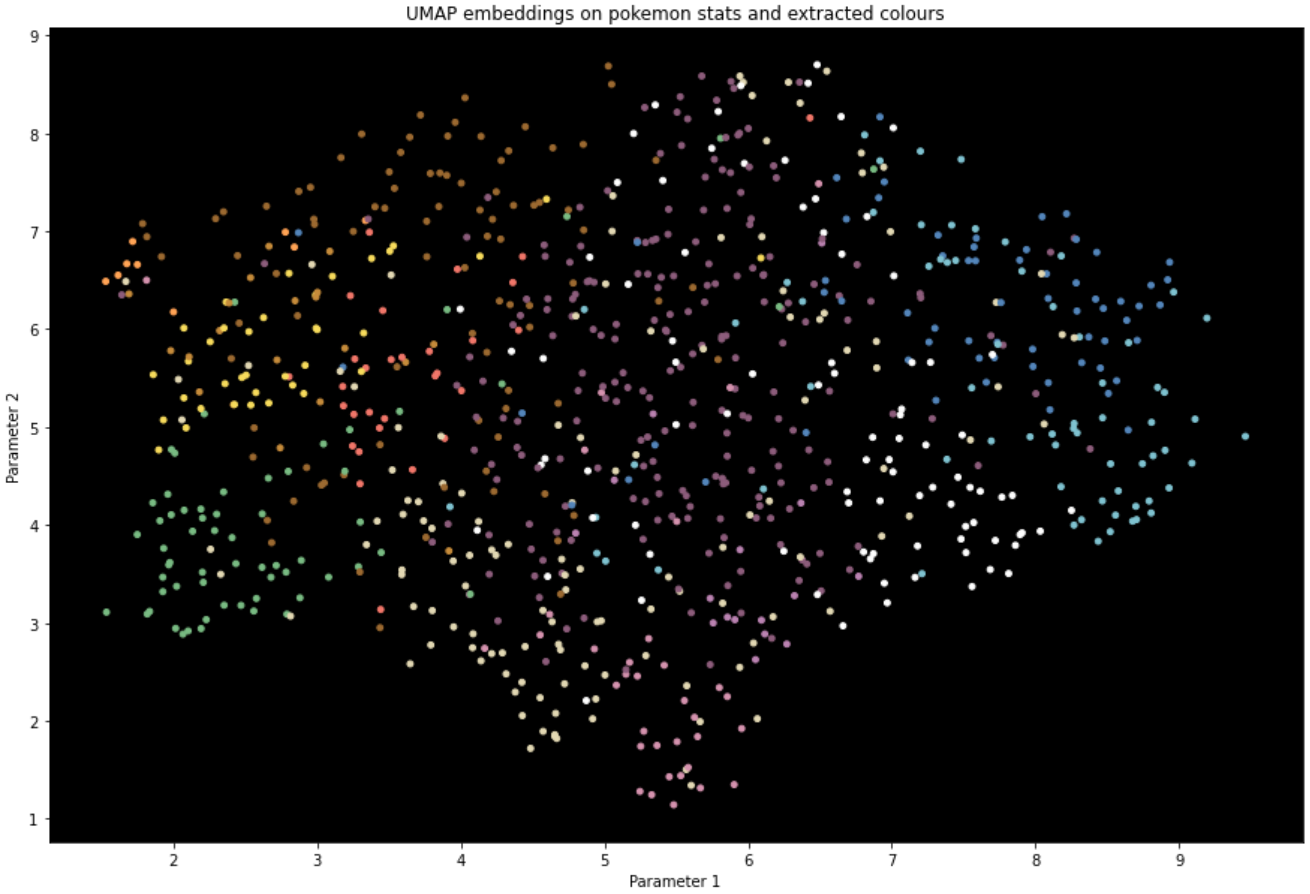
UMAP

Preprocessing:

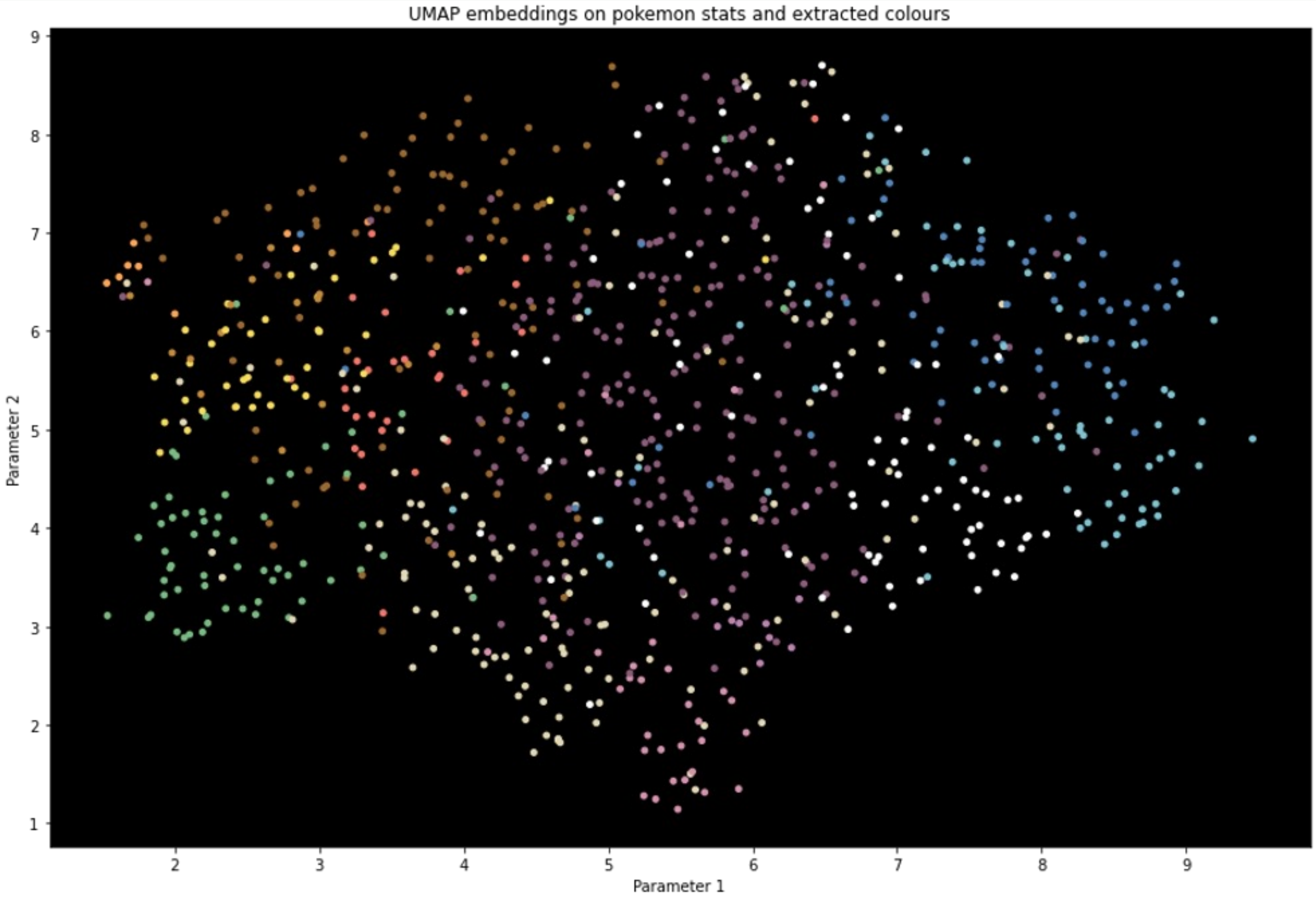
- Classified each pixel as a colour, based on proximity to list of RGB colours and removed all counts of 'black'
- Encoded Pokemon abilities 1-3

Input:

- Number of pixels in each color
- Shapes
- Stats
- Encoded abilities

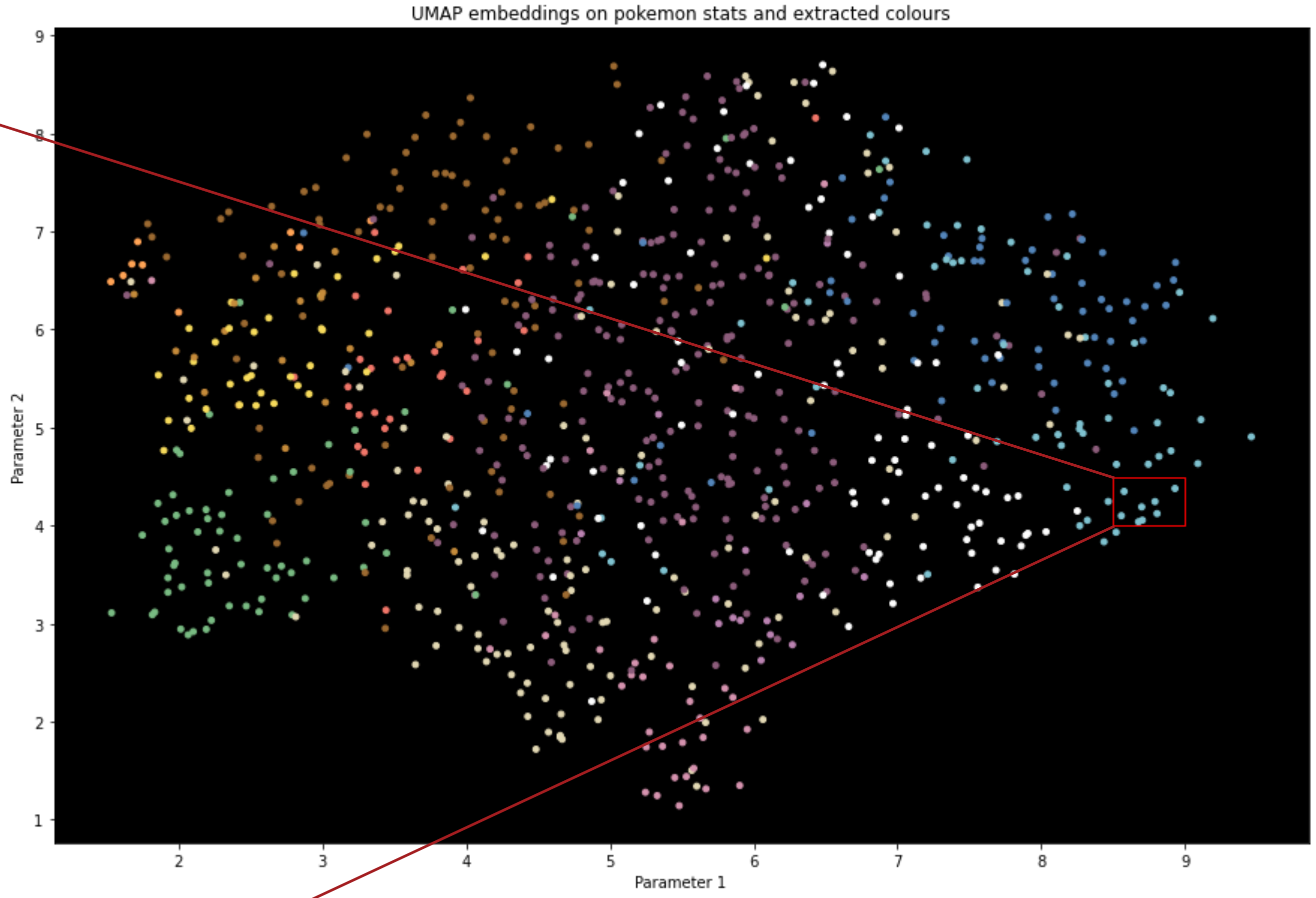


Dot-colour = Most occurring colour in image



Dot-colour = Most occurring colour in image

Clustering



Dot-colour = Most occurring colour in image

Clustering

Model:

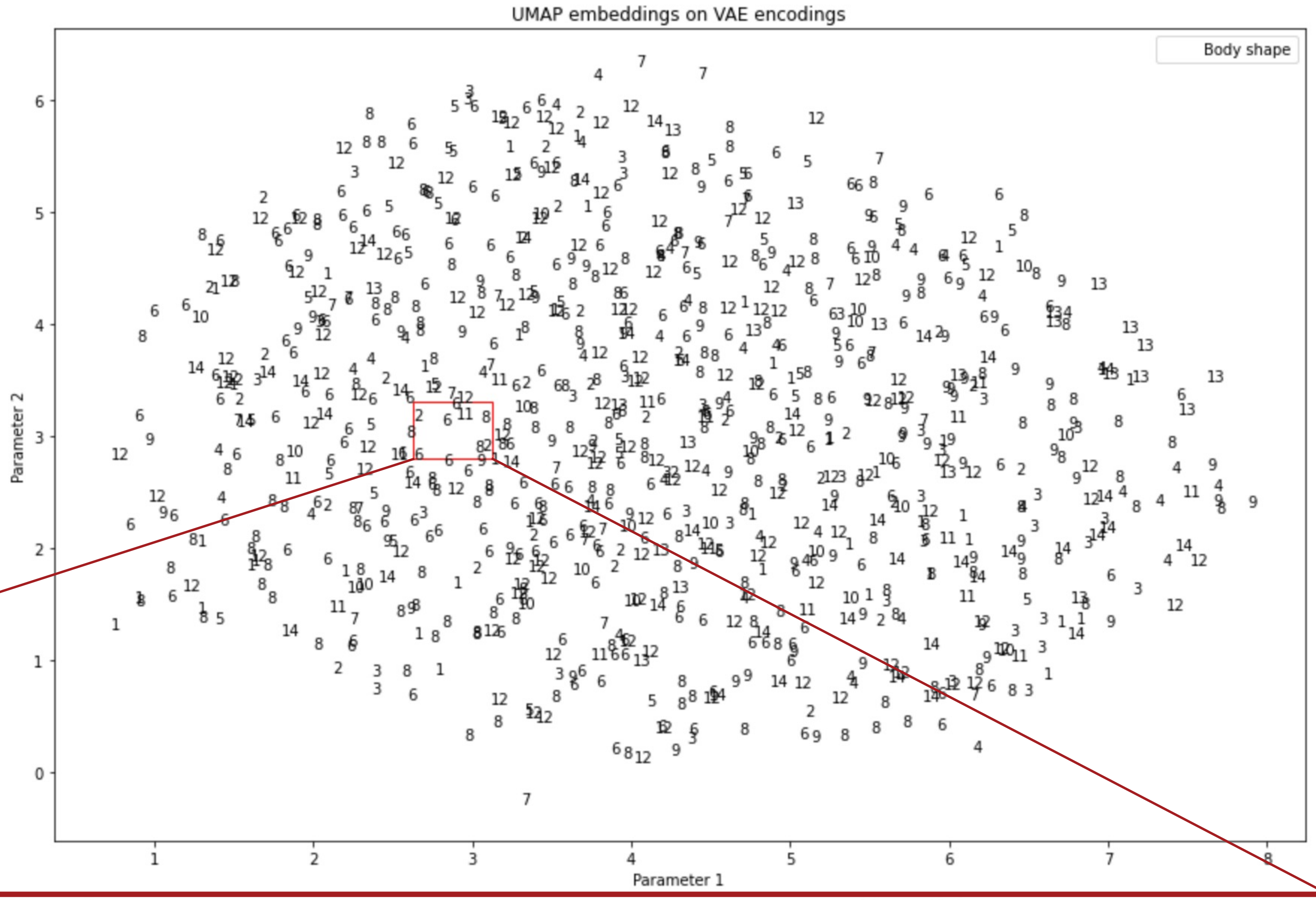
- UMAP

Preprocessing:

- VAE model encodings of each pokémon image

Input:

- 512D latent vectors



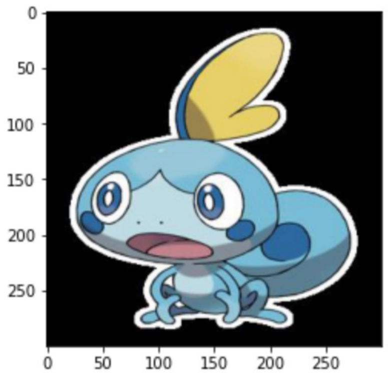
HUMAN BENCHMARK

Human benchmark

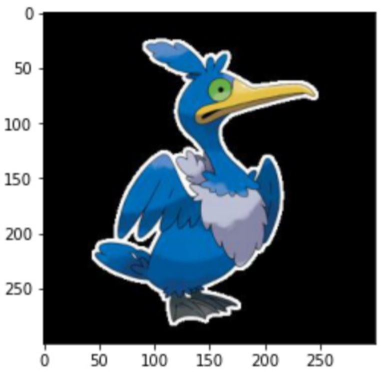
- How good is a human at this task?
- Construct fair testing for humans:
  - Allowed to look at all previous pokemon and their types (training set)
  - Allowed to see each test image once and predict a type

Human model	Accuracy
Corner	58.5 %
Astrid (Friend)	57.3 %
Azzura (TA)	53.6 %
Julie (Girlfriend)	53.6 %
Frederik	50.0 %
Random guessing	5.5 % (1/18)

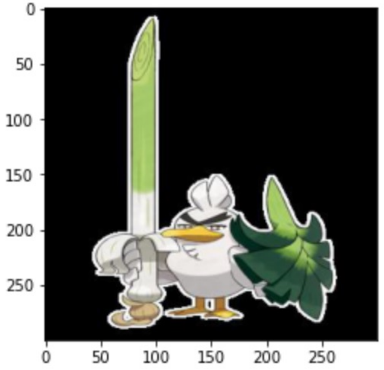
82x



Pokemon: Sobble
Actual type: Water
Corner predicted type: Water



Pokemon: Cramorant
Actual type: Flying
Corner predicted type: Flying



Pokemon: Sirfetch'd
Actual type: Fighting
Corner predicted type: Normal

...

Human benchmark

Human model	Accuracy
Kristoffer (Friend)	50.0 %

- Tested in random order – avoids unfair information transfer between evolutions
- Conclusion: Hard task even for humans
  - 11 pokémon were not guessed by any human
  - Many pokémon requires abstract reasoning to predict

Pokémon: Clobbopus
Actual type: Fighting
Predicted types: ['Water' 'Water' 'Water' 'Psychic' 'Psychic' 'Water']

Pokémon: Hatenna
Actual type: Psychic
Predicted types: ['Fairy' 'Fairy' 'Fairy' 'Fairy' 'Normal' 'Fairy']

Pokémon: Hattrem
Actual type: Psychic
Predicted types: ['Fairy' 'Fairy' 'Fairy' 'Fairy' 'Normal' 'Fairy']

Pokémon: Perrserker
Actual type: Steel
Predicted types: ['Fighting' 'Normal' 'Fighting' 'Ground' 'Ground' 'Dark']

Pokémon: Eternatus
Actual type: Poison
Predicted types: ['Dragon' 'Dragon' 'Dark' 'Dragon' 'Ghost' 'Dragon']

Pokémon: Regidrago
Actual type: Dragon
Predicted types: ['Dark' 'Ghost' 'Dark' 'Dark' 'Dark' 'Steel']

Pokémon: Runerigus
Actual type: Ground
Predicted types: ['Ghost' 'Rock' 'Rock' 'Psychic' 'Rock' 'Dragon']

Pokémon: Falinks
Actual type: Fighting
Predicted types: ['Bug' 'Bug' 'Bug' 'Electric' 'Bug' 'Bug']

Pokémon: Pincurchin
Actual type: Electric
Predicted types: ['Poison' 'Poison' 'Water' 'Dark' 'Ground' 'Dark']

Pokémon: Cufant
Actual type: Steel
Predicted types: ['Ground' 'Normal' 'Grass' 'Normal' 'Normal' 'Normal']

Pokémon: Arctozolt
Actual type: Electric
Predicted types: ['Ice' 'Dragon' 'Dragon' 'Ice' 'Ice' 'Water']

XGBOOST

Predicting the type

Model	Accuracy	F1-score
Human average	54.6 %	52.5 %
XGBoost on stats	17.0 %	13.9 %

Model type:

- XGBoost Classifier

Preprocessing:

- None

Input:

- Stats

Evaluation:

- Poor performance

ID	Name	Type1	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed
1	Bulbasaur	Grass	318	45	49	49	65	65	45

Predicting the type

Model	Accuracy	F1-score
Human average	54.6 %	52.5 %
XGBoost on stats	17.0 %	13.9 %
XGBoost on extracted base-colours	17.0 %	13.7 %

Model type:

- XGBoost Classifier

Preprocessing:

- Classified each pixel as a colour, based on proximity to list of RGB colours

Input:

- Number of pixels in each colour

Evaluation:

- Poor performance
- Colours too bright in comparison

```
colors = {'red': (255,0,0),
          'green': (0,255,0),
          'blue': (0,0,255),
          'yellow': (255,255,0),
          'orange': (255,127,0),
          'white': (255,255,255),
          'black': (0,0,0),
          'gray': (127,127,127),
          'pink': (255,127,127),
          'purple': (127,0,255)}
```

Predicting the type

Model	Accuracy	F1-score
Human average	54.6 %	52.5 %
XGBoost on stats	17.0 %	13.9 %
XGBoost on extracted base-colours	17.0 %	13.7 %
XGBoost on hand-picked colours	26.8 %	25.0 %

Model type:

- XGBoost Classifier

Preprocessing:

- Classified each pixel as a colour, based on proximity to list of RGB colours

Input:

- Number of pixels in each colour

Evaluation:

- Big improvement
- Colours match pokémons better

```
colors = {'red': (227,125,111),
          'green': (136,185,136),
          'light_blue': (142,194,207),
          'dark_blue': (94,134,183),
          'yellow': (244,219,113),
          'orange': (249,168,101),
          'light_brown': (190,141,74),
          'dark_brown': (149,107,59),
          'white': (255,255,255),
          'black': (0,0,0),
          'gray': (226,217,184),
          'pink': (204,150,174),
          'light_purple': (179,134,175),
          'dark_purple': (134,96,122)}
```

Predicting the type

Model	Accuracy	F1-score
Human average	54.6 %	52.5 %
XGBoost on stats	17.0 %	13.9 %
XGBoost on extracted base-colours	17.0 %	13.7 %
XGBoost on hand-picked colours	26.8 %	25.0 %
XGBoost on type-colours	23.2 %	19.8 %

Model type:

- XGBoost Classifier

Preprocessing:

- Classified each pixel as a colour, based on proximity to list of RGB colours

Input:

- Number of pixels in each colour

Evaluation:

- Preprocessing worse than hand-picked colours

```
colors = {"bug_colour":   [168, 185, 32], #'Bug'
          "dark_colour":   [112, 88, 72], #'Dark'
          "dragon_colour":   [112, 56, 255], #'Dragon'
          "electric_colour":   [248, 209, 255], #'Electric'
          "fairy_colour":   [240, 182, 188], #'Fairy'
          "fighting_colour":   [192, 48, 40], #'Fighting'
          "fire_colour":   [239, 128, 48], #'Fire'
          "flying_colour":   [168, 144, 240], #'Flying'
          "ghost_colour":   [112, 88, 255], #'Ghost'
          "grass_colour":   [120, 200, 79], #'Grass'
          "ground_colour": [224, 192, 104], # 'Ground'
          "ice_colour": [152, 216, 216], # 'Ice'
          "normal_colour": [168, 168, 120], # 'Normal'
          "poison_colour": [159, 64, 159], # 'Poison'
          "psychic_colour": [248, 88, 136], # 'Psychic'
          "rock_colour": [184, 160, 57], # 'Rock'
          "steel_colour": [184, 184, 208], # 'Steel'
          "water_colour": [103, 144, 240]} # 'Water'
```

Predicting the type

Model type:

- XGBoost Classifier

Preprocessing:

- Classified each pixel as a colour, based on proximity to list of RGB colours

Input:

- Number of pixels in each colour

Evaluation:

- Preprocessing worse than hand-picked colours

Model	Accuracy	F1-score
Human average	54.6 %	52.5 %
XGBoost on stats	17.0 %	13.9 %
XGBoost on extracted base-colours	17.0 %	13.7 %
XGBoost on hand-picked colours	26.8 %	25.0 %
XGBoost on type-colours	23.2 %	19.8 %

Name: Troels
Predicted types: Fairy and Rock

CNN

Predicting the type

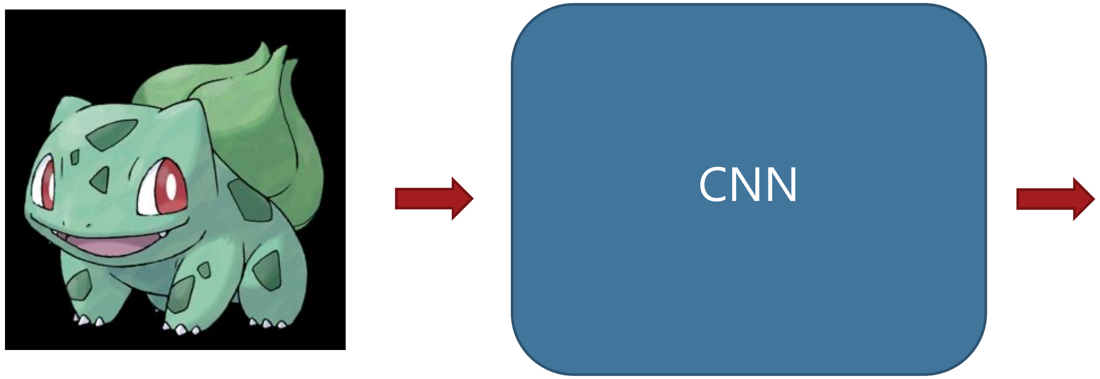
CNN from scratch

Pretrained CNNs

- Feature extraction
- Full model

Trained on

- Only images
- Images + stats, shapes, colors, etc.



- ○ Water
- ○ Fire
- ● Grass
- ○ Bug
- ○ Darm
- ○ Psychic
- ○ Electric
- ○ Fairy
- ○ Fighting
- ○ Normal
- ○ Flying
- ○ Ghost
- ○ Ground
- ○ Ice
- ○ Poison
- ○ Dragon
- ○ Rock
- ○ Steel

Predicting the type

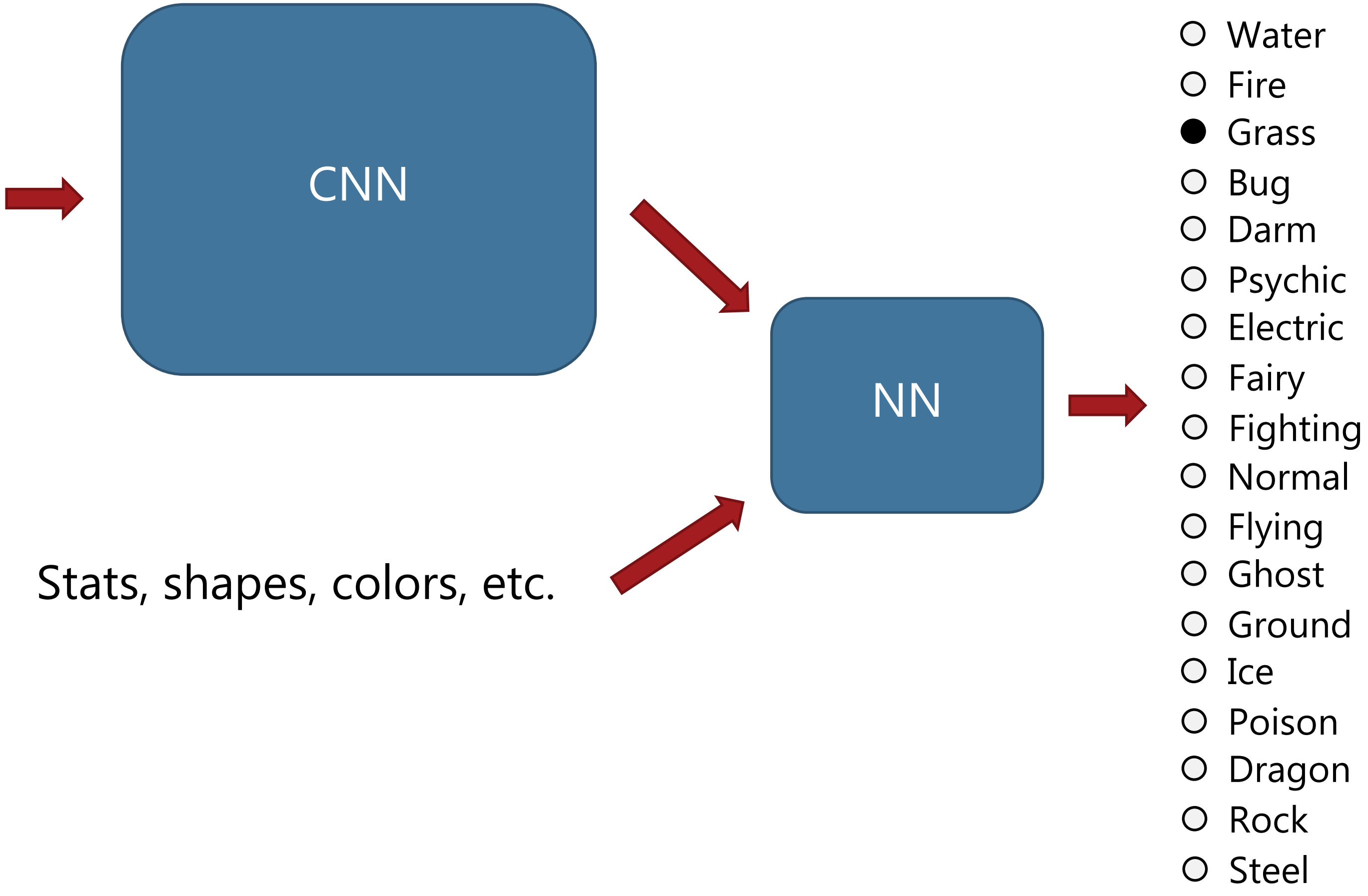
CNN from scratch

Pretrained CNNs

- Feature extraction
- Full model

Trained on

- Only images
- Images + stats, shapes, colors, etc.



Predicting the type

Model type:

- CNN from scratch

Input:

- Only images

Preprocessing:

- Normalize pixel values

Evaluation:

- Not great

Model	Accuracy
Human average	54.6 %
CNN from scratch	15.9 %

```
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.ConvLayer1 = nn.Sequential(
            nn.Conv2d(3, 8, 3),
            nn.Conv2d(8, 16, 3),
            nn.MaxPool2d(2),
            nn.ReLU()
        )
        self.ConvLayer2 = nn.Sequential(
            nn.Conv2d(16, 32, 5),
            nn.Conv2d(32, 32, 3),
            nn.MaxPool2d(4),
            nn.ReLU()
        )
        self.ConvLayer3 = nn.Sequential(
            nn.Conv2d(32, 64, 3),
            nn.Conv2d(64, 64, 5),
            nn.MaxPool2d(2),
            nn.ReLU()
        )
        self.ConvLayer4 = nn.Sequential(
            nn.Conv2d(64, 128, 5),
            nn.Conv2d(128, 128, 3),
            nn.MaxPool2d(2),
            nn.ReLU()
        )
        self.Lin1 = nn.Linear(2048, 1500)
        self.Lin2 = nn.Linear(1500, 150)
        self.Lin3 = nn.Linear(150, 18)

    def forward(self, x):
        x = self.ConvLayer1(x)
        x = self.ConvLayer2(x)
        x = self.ConvLayer3(x)
        x = self.ConvLayer4(x)
        x = x.view(x.size(0), -1)
        x = self.Lin1(x)
        x = self.Lin2(x)
        x = self.Lin3(x)
        return nn.functional.log_softmax(x, dim = 1)
```

Predicting the type

Model	Accuracy
Human average	54.6 %
CNN from scratch	15.9 %
CNN feature extract ResNet18	25.6 %

Model type:

- ResNet18 feature extract

Input:

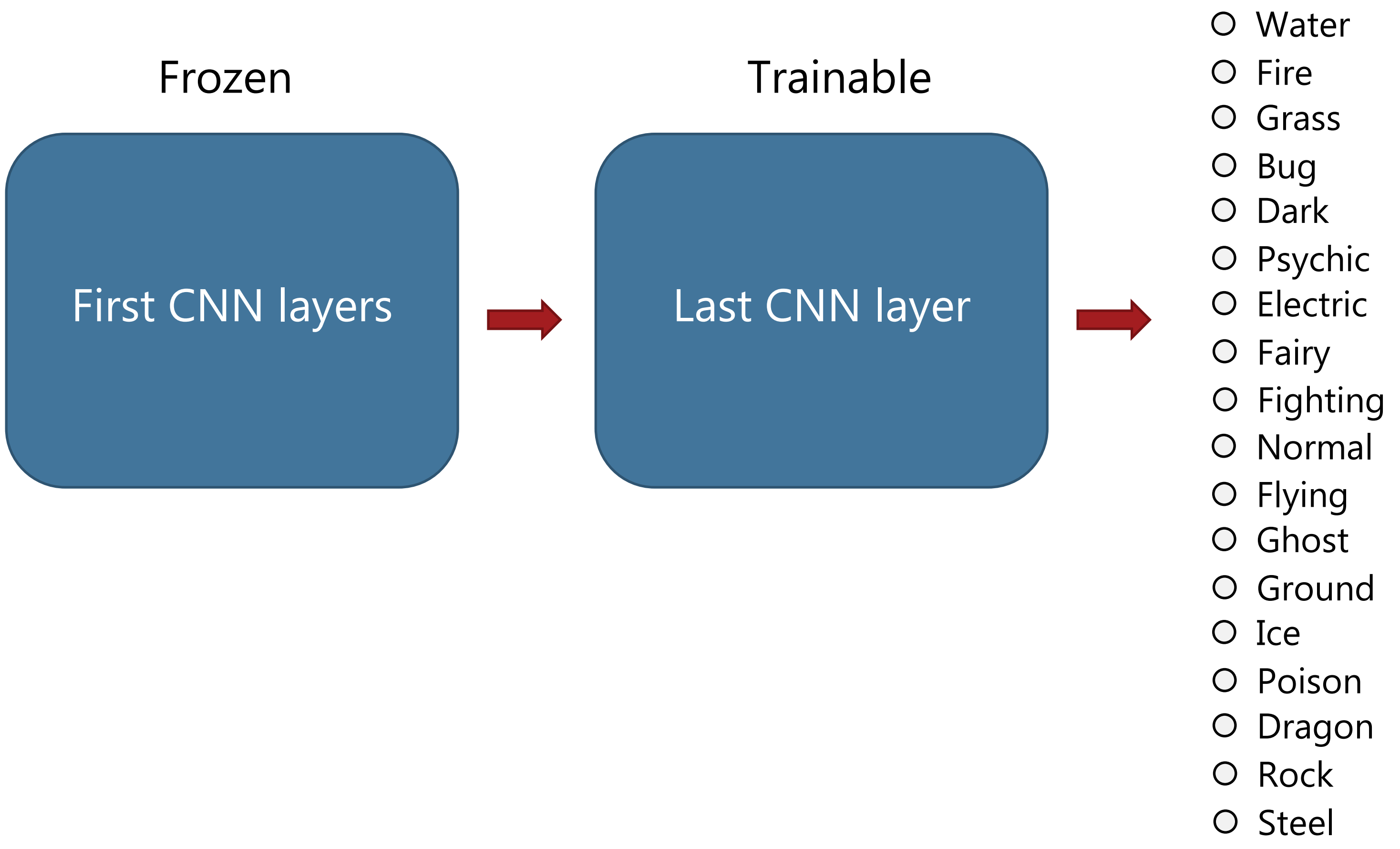
- Only images

Preprocessing:

- Normalize pixel values

Evaluation:

- Pretty good



Predicting the type

Model	Accuracy
Human average	54.6 %
CNN from scratch	15.9 %
CNN feature extract ResNet18	25.6 %
CNN full ResNet18	28.1 %

Model type:

- Modified ResNet18

Input:

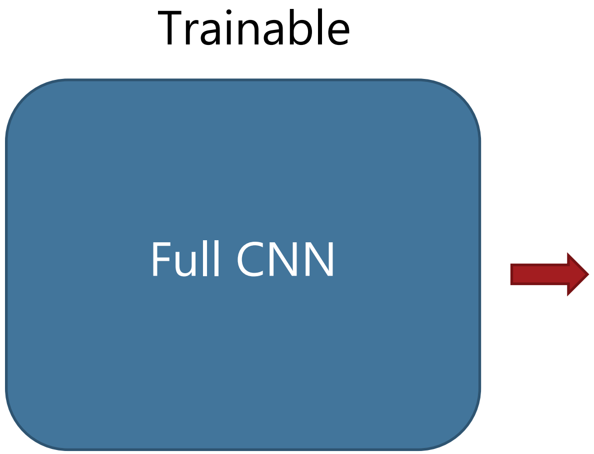
- Only images

Preprocessing:

- Normalize pixel values

Evaluation:

- Even better



- Fire
- Grass
- Bug
- Dark
- Psychic
- Electric
- Fairy
- Fighting
- Normal
- Flying
- Ghost
- Ground
- Ice
- Poison
- Dragon
- Rock
- Steel

Predicting the type

Model type:

- Modified ResNet18

Input:

- Only images

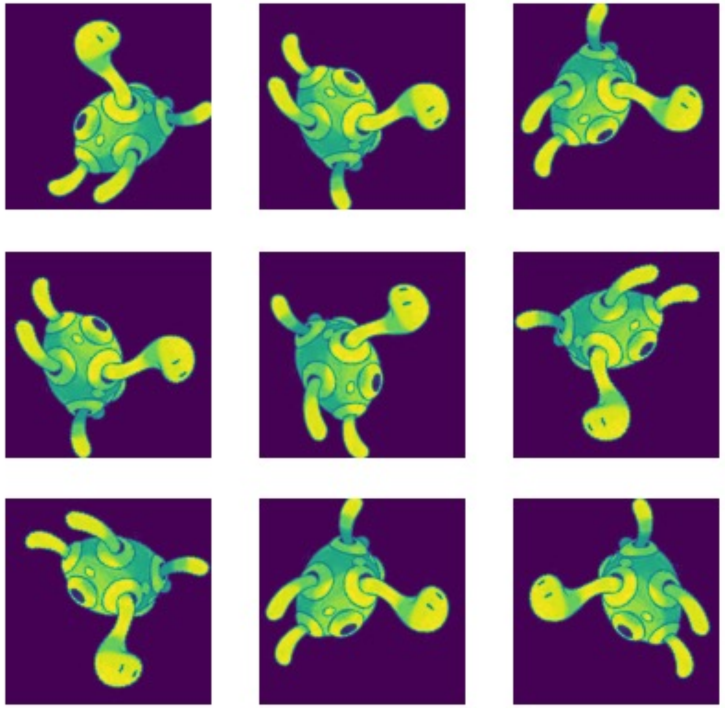
Preprocessing:

- Normalize pixel values
- Augment train images

Evaluation:

- Very good model

Model	Accuracy
Human average	54.6 %
CNN from scratch	15.9 %
CNN feature extract ResNet18	25.6 %
CNN full ResNet18	28.1 %
CNN full ResNet18 + augment	34.2 %



Predicting the type

Model type:

- Modified ResNet18

Input:

- Only images
- Including stats, shapes, colors

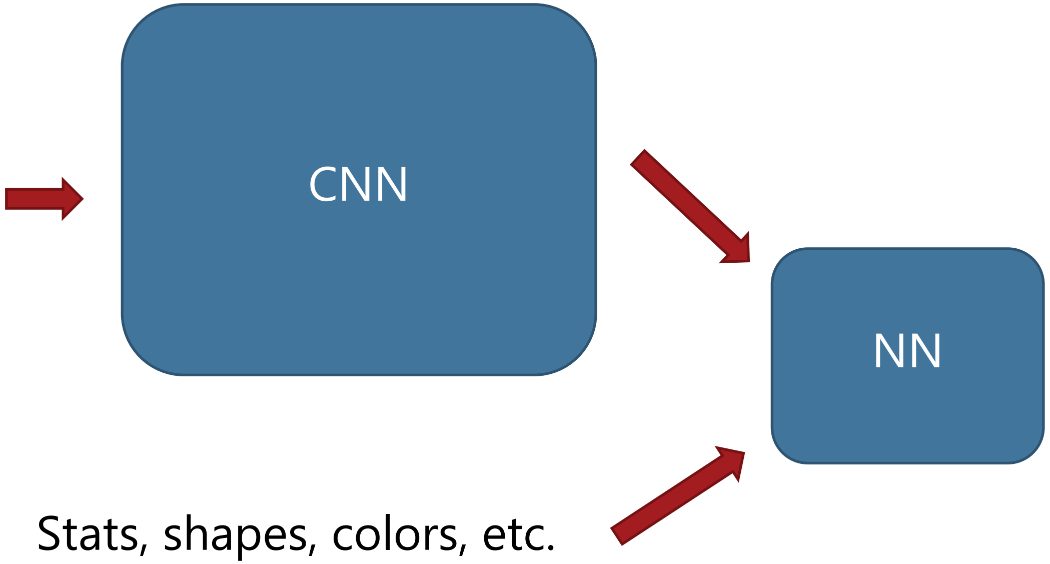
Preprocessing:

- Normalize pixel values
- Augment train images

Evaluation:

- Same as without scalar inputs

Model	Accuracy
Human average	54.6 %
CNN from scratch	15.9 %
CNN feature extract ResNet18	25.6 %
CNN full ResNet18	28.1 %
CNN full ResNet18 + augment	34.2 %
CNN full ResNet18 + augment + scalar inputs	34.2 %



Predicting the type

Model type:

- Modified ResNet18

Input:

- Only images
- Including stats, shapes, colors

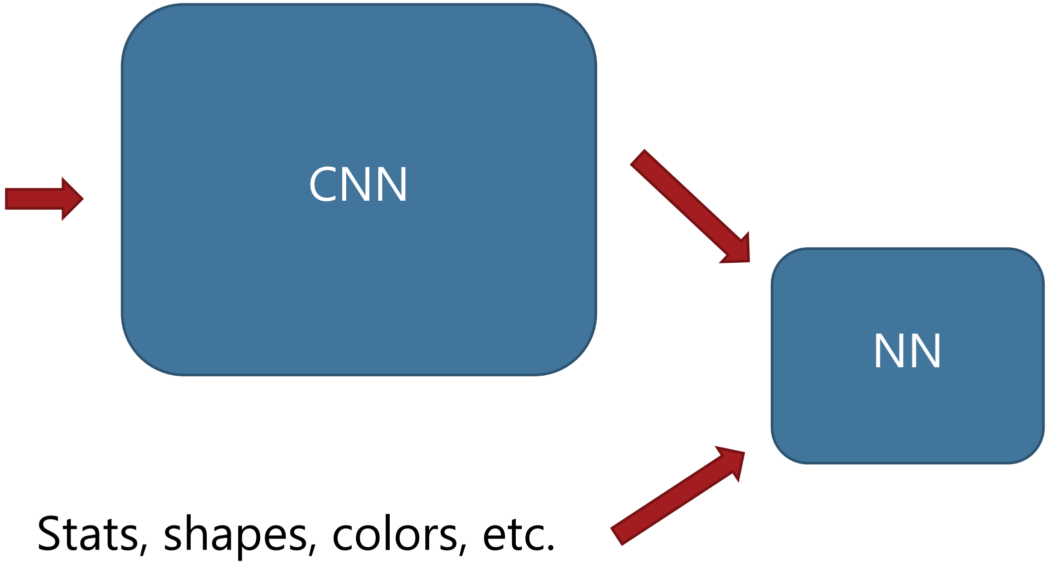
Preprocessing:

- Normalize pixel values
- Augment train images

Evaluation:

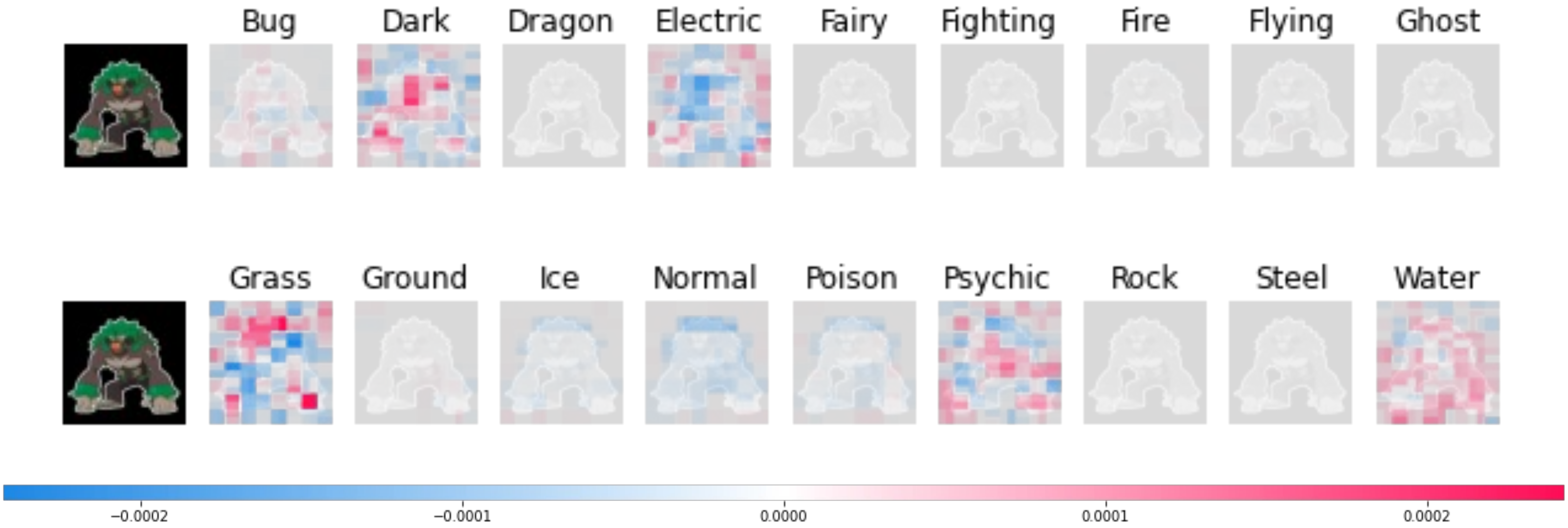
- Same as without scalar inputs

Model	Accuracy
Human average	54.6 %
CNN from scratch	15.9 %
CNN feature extract ResNet18	25.6 %
CNN full ResNet18 + scalar inputs	34.2 %
CNN full ResNet18 + augment	34.2 %
CNN full ResNet18 + augment + scalar inputs	34.2 %



Explaining CNN predictions

- Shap values



VAE

Reconstructing Pokémon images

Variational Autoencoder

Model structure: 87 M parameters

- Encoder
  - 4 convolutional layers
  - 3 dense layers
- Latent space (2, 3, 4, 128, 256, 512 dim)
- Decoder
  - 1 dense layer
  - 5 convolutional layers

Data:

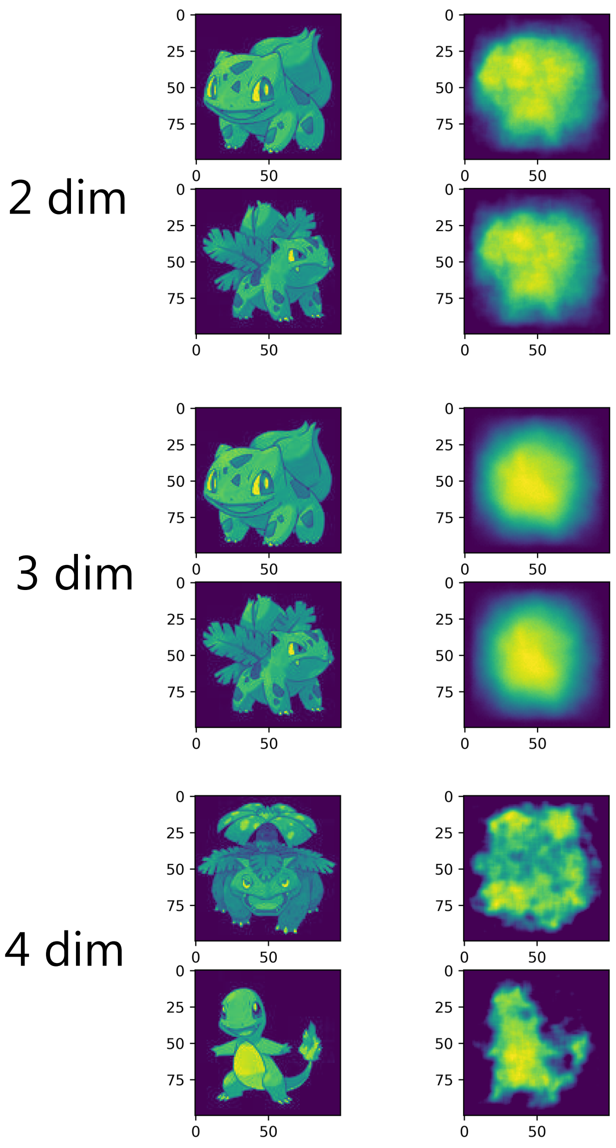
- 100x100 Greyscale images

```
Model: "encoder"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 encoder input (InputLayer)     [(None, 100, 100, 1  0           []
                                )]

 conv2d (Conv2D)                (None, 100, 100, 12  3328        ['encoder input[0][0]']
                                8)

 conv2d_1 (Conv2D)              (None, 50, 50, 64)   73792       ['conv2d[0][0]']

 conv2d_2 (Conv2D)              (None, 50, 50, 64)   36928       ['conv2d_1[0][0]']

 conv2d_3 (Conv2D)              (None, 50, 50, 64)   36928       ['conv2d_2[0][0]']

 flatten (Flatten)              (None, 160000)       0           ['conv2d_3[0][0]']

 dense (Dense)                  (None, 32)           5120032     ['flatten[0][0]']

 Z-mean (Dense)                 (None, 512)          16896       ['dense[0][0]']

 Z-logvariance (Dense)          (None, 512)          16896       ['dense[0][0]']

 latent-space (Lambda)          (None, 512)          0           ['Z-mean[0][0]',
                                                                  'Z-logvariance[0][0]']

==================================================================================================
Total params: 5,304,800
Trainable params: 5,304,800
Non-trainable params: 0
__________________________________________________________________________________________________
```

```
Model: "decoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 decoder-input (InputLayer)  [(None, 512)]             0

 dense_1 (Dense)             (None, 160000)            82080000

 reshape (Reshape)           (None, 50, 50, 64)        0

 conv2d_transpose (Conv2DTra  (None, 50, 50, 64)       36928
 nspose)

 conv2d_transpose_1 (Conv2DT  (None, 50, 50, 64)       36928
 ranspose)

 conv2d_transpose_2 (Conv2DT  (None, 100, 100, 64)     36928
 ranspose)

 conv2d_transpose_3 (Conv2DT  (None, 100, 100, 128)    204928
 ranspose)

 conv2d_transpose_4 (Conv2DT  (None, 100, 100, 1)      3201
 ranspose)

=================================================================
Total params: 82,398,913
Trainable params: 82,398,913
Non-trainable params: 0
_________________________________________________________________
```

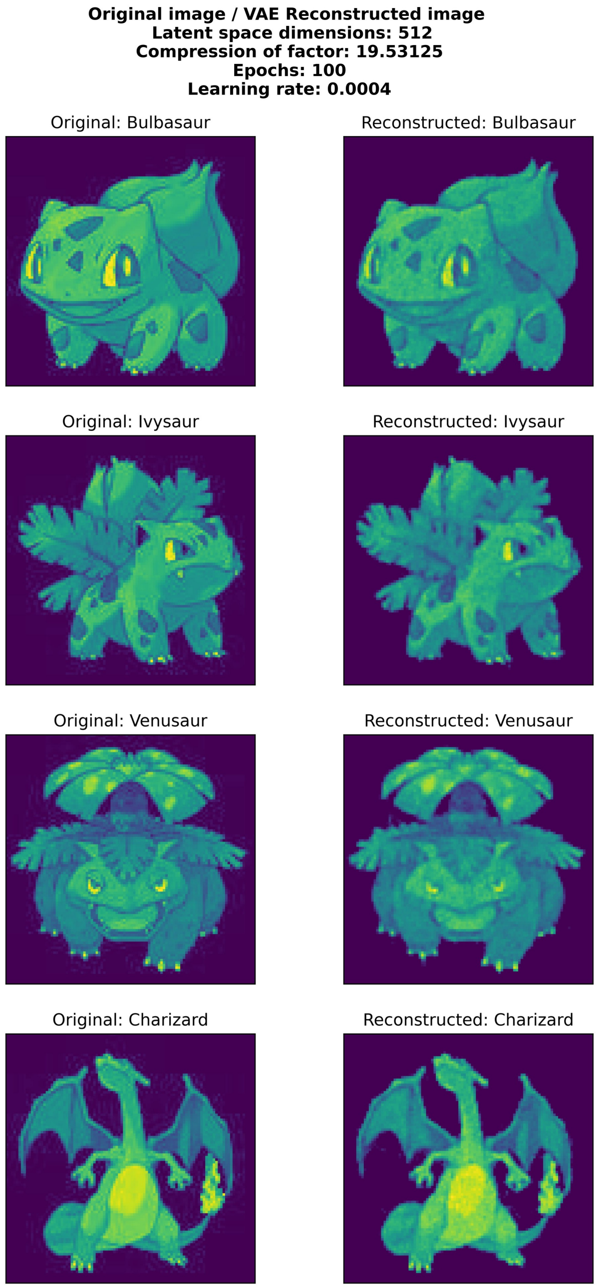
Reconstructing Pokémon images

- How large does the latent space need to be?
- 2-4 is much too small to encode detailed images



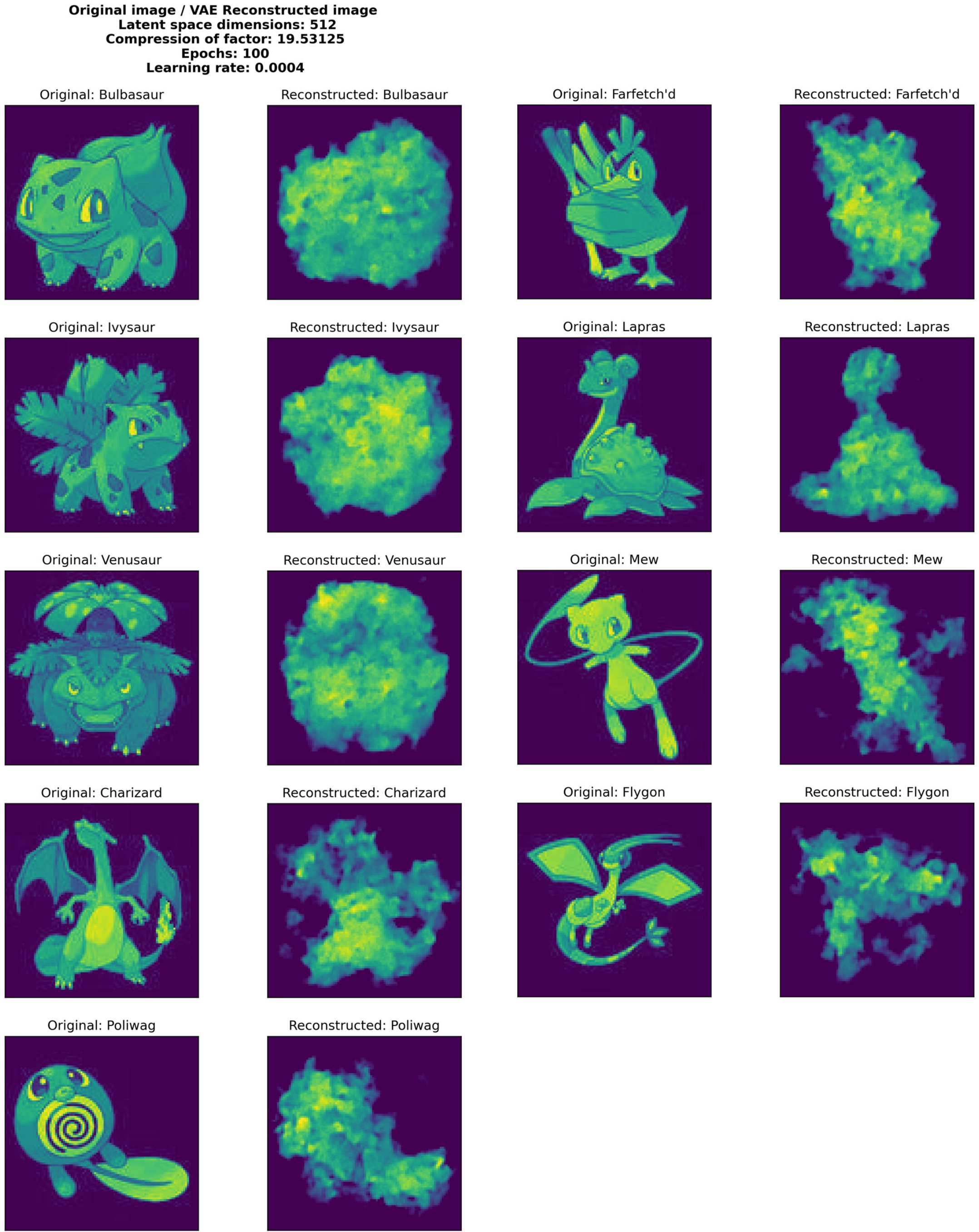
Reconstructing Pokémon images

- How large does the latent space need to be?
- This is reconstructing images from the training set. Always possible with a large model



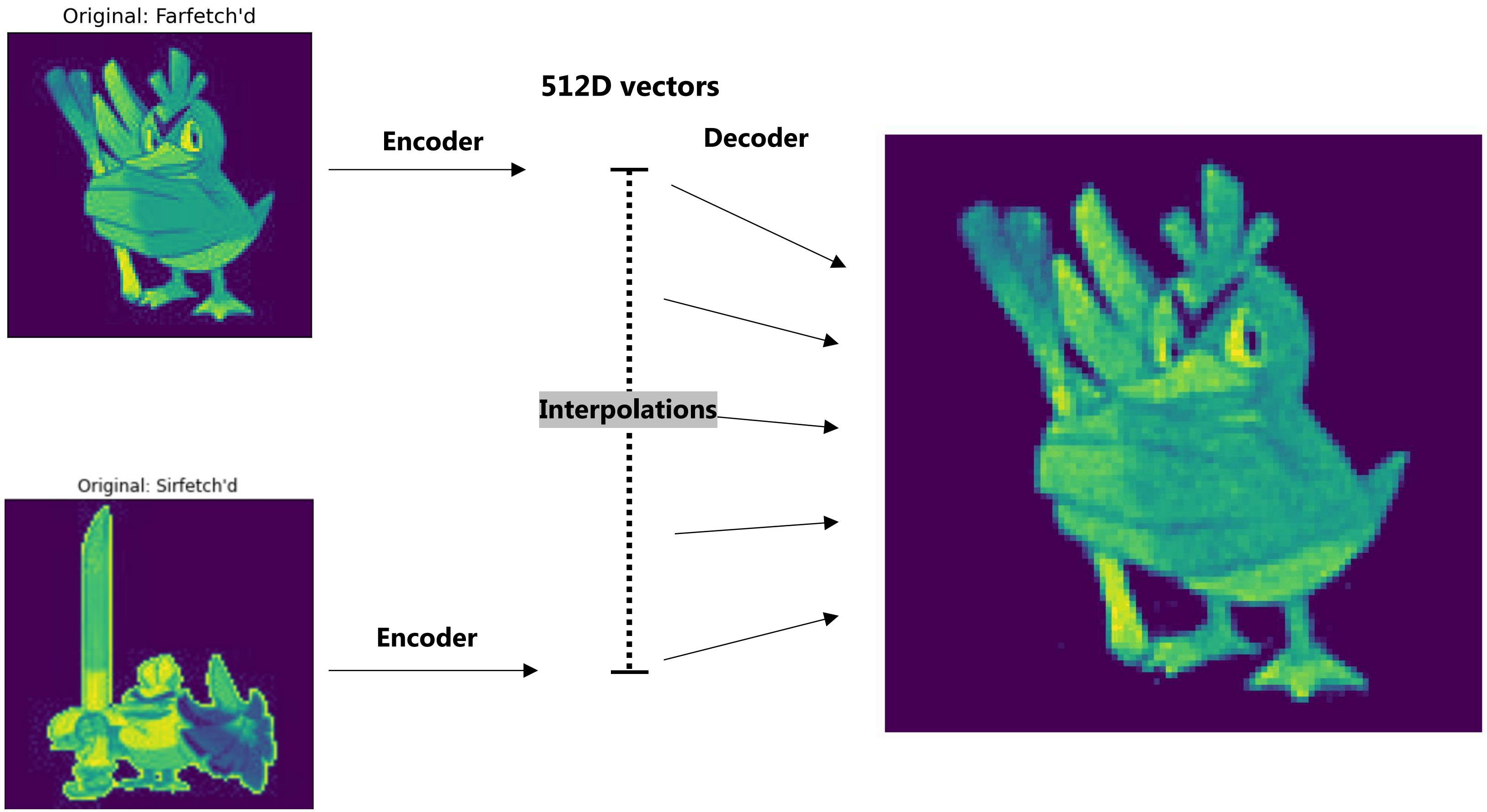
Reconstructing Pokémon images

- Is is possible to reconstruct the pokémon just shown, without training on them?
- Not really, since there aren't 1000 Pokémon very similar to Bulbasaur, etc. to fill the pokémon space
- Pokémon are inherently different
- General shape is reconstructable



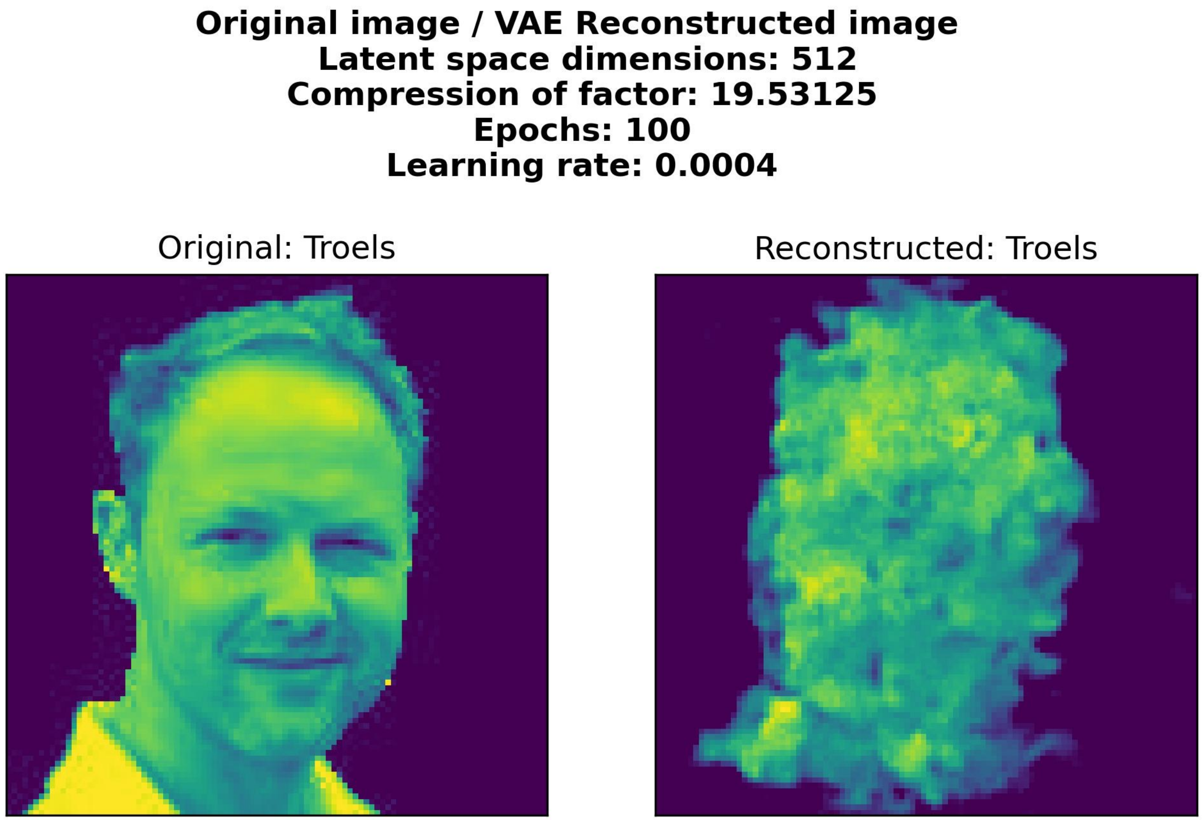
Reconstructing Pokémon images

- Let's explore the 512D space!
- Moving from Pokémon to Pokémon encoding in the latent space
- Decoding from interpolations



Reconstructing Pokémon images

- Could it reconstruct Troels?
  - No, but the general shape is there
- Which Pokémon are the closest to Troels in the 512D latent space?



DISCUSSION AND CONCLUSION

Discussion

- Goal: Beat human performance
- Result: Not quite reached

Model	Accuracy
Human average	54.6 %
Best model	34.2 %

Some explanations:

- Some Pokémon might just be unpredictable
- Only ca. 900 datapoints
- Difference in preconception

Discussion – Human preconception advantages

Human preconception

- Pokémon knowledge
  - 3 starter Pokémon (grass, fire, water) with same-type evolutions
  - Same type through evolution
  - Systematic Pokédex order
  - Flying is recessive, normal is exclusive
- Preconception of shapes
  - Bugs, lightning, wings, leaves, flames
- Perceive black as non-background

Model data

- Utilizes large amounts of stats
- Un-biased pattern recognition

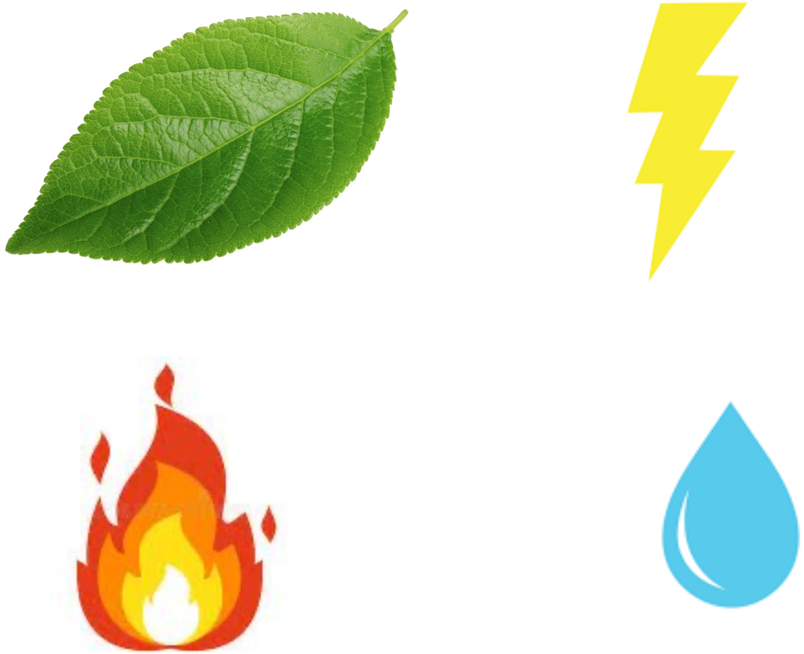
Improvements

Natural Language Processing

- Pokémon names

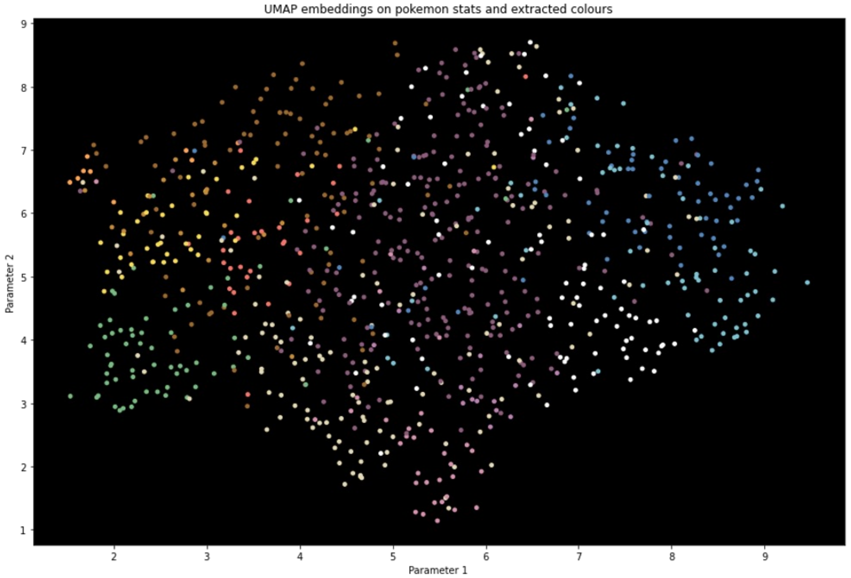
Ivysaur

More related images to Pokémon types in image dataset

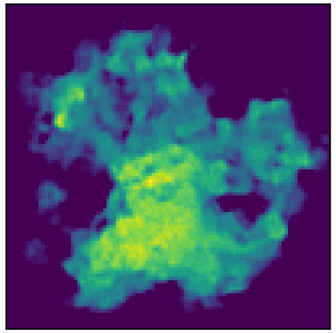


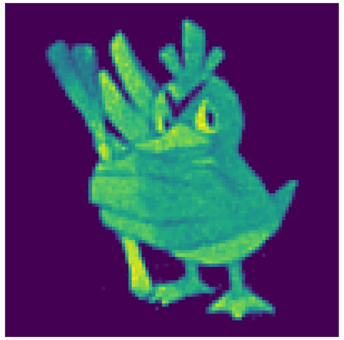
Conclusion

Clustering



VAE





XGBoost

Model	Accuracy
Human average	54.6 %
XGBoost on stats	17.0 %
XGBoost on extracted base-colours	17.0 %
XGBoost on hand-picked colours	**26.8 %**
XGBoost on type-colours	23.2 %

CNNs

Model	Accuracy
Human average	54.6 %
CNN from scratch	15.9 %
CNN feature extract ResNet18	25.6 %
CNN full ResNet18 + scalar inputs	**34.2 %**
CNN full ResNet18 + augment	**34.2 %**
CNN full ResNet18 + augment + scalar inputs	**34.2 %**

Appendix

Details on the Pokémon Project by Frederik, Frederik, Marcus and Tobias

Tree Based model

XGBoost Classifier (standard)

- Learning rate: 0.1
- Maximum tree depth: 5
- Number of boosting rounds: 500
- Number of threads: 6
- Objective: 'multi:softmax'

Run with early stopping

XGBoost Classifier (random search optimized)

- Learning rate: 0.02079
- Maximum tree depth: 4
- Number of boosting rounds: 500
- Number of threads: 8
- Objective: 'multi:softmax'

Run with early stopping rounds: 10

Subsample: 0.941
colsample_bytree: 0.927
colsample_bylevel: 0.907

Tree Based model

Random search parameters:

- Max tree depth: 3-10
- Learning rate: 0.01-0.3 (step: 0.0001)
- Subsample: 0.5-1.0 (step: 0.001)
- colsample_bytree: 0.4-1.0 (step: 0.001)
- colsample_bylevel: 0.4-1.0 (step: 0.001)
- Number of boosting rounds: 150
- Number of threads: 6
- Metric: logloss

Predicting the type

Model	Accuracy	F1-score
Human average	54.6 %	52.5 %
XGBoost on stats	17.0 %	13.9 %
XGBoost on extracted base-colours	17.0 %	13.7 %

Model type:

- XGBoost Classifier

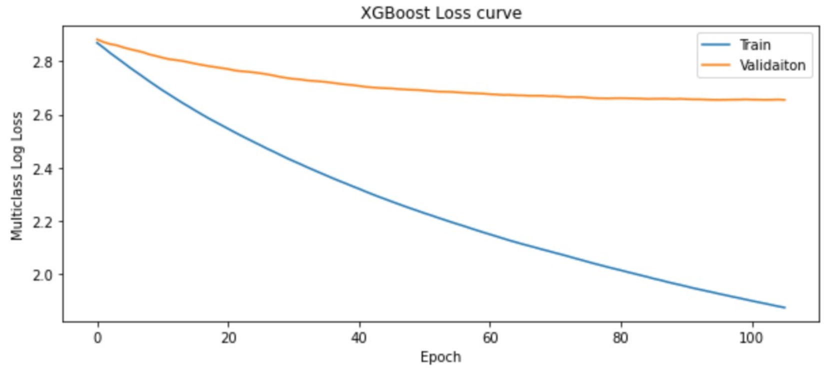
Preprocessing:

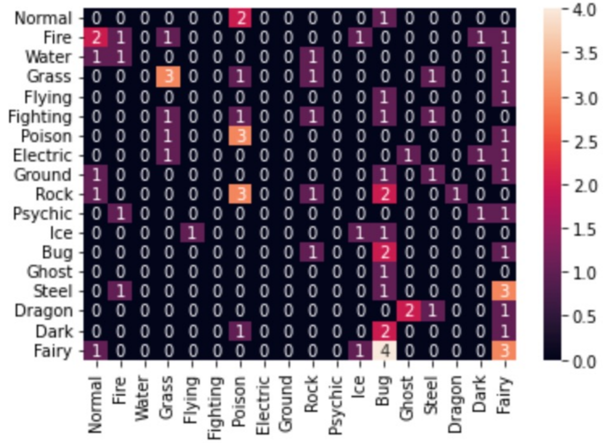
- Classified each pixel as a colour, based on proximity to list of RGB colours

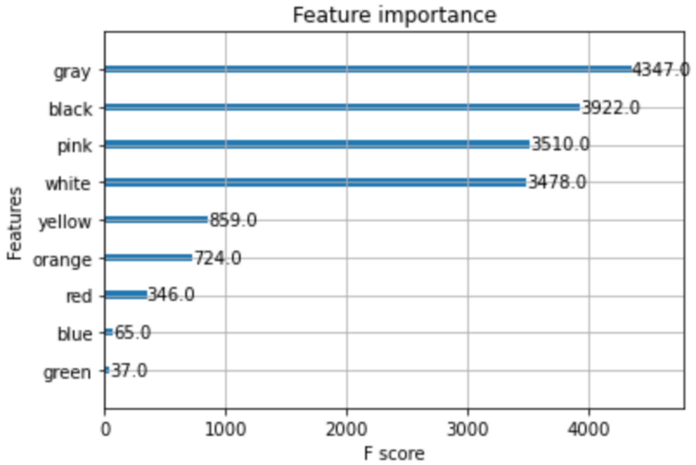
Input:

- Number of pixels in each colour

Evaluation:







Predicting the type

Model	Accuracy	F1-score
Human average	54.6 %	52.5 %
XGBoost on stats	17.0 %	13.9 %
XGBoost on extracted base-colours	17.0 %	13.7 %
XGBoost on hand-picked colours	26.8 %	25.0 %

Model type:

- XGBoost Classifier

Preprocessing:

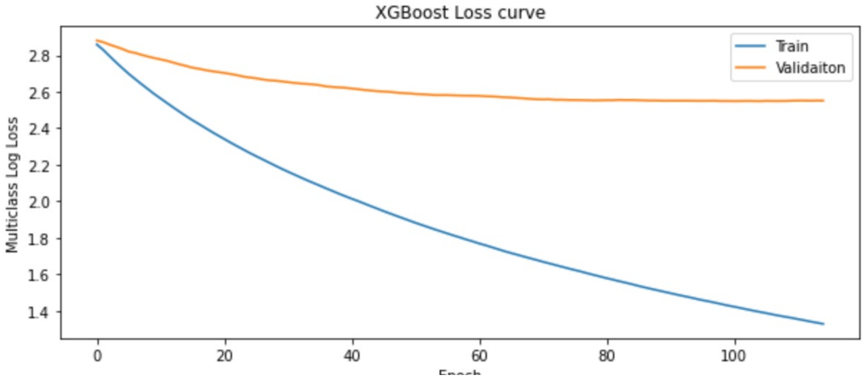
- Classified each pixel as a colour, based on proximity to list of RGB colours

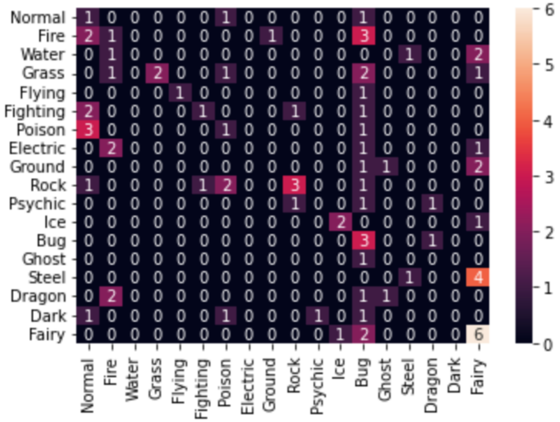
Input:

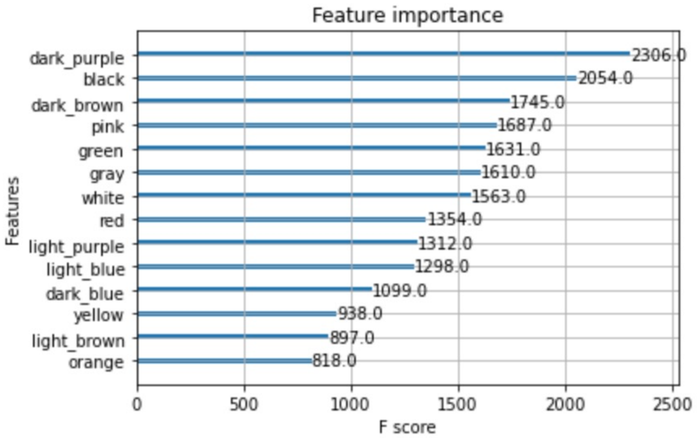
- Number of pixels in each colour

Evaluation:

- Better performance







Predicting the type

Model type:

- XGBoost Classifier

Preprocessing:

- Classified each pixel as a colour, based on proximity to list of RGB colours

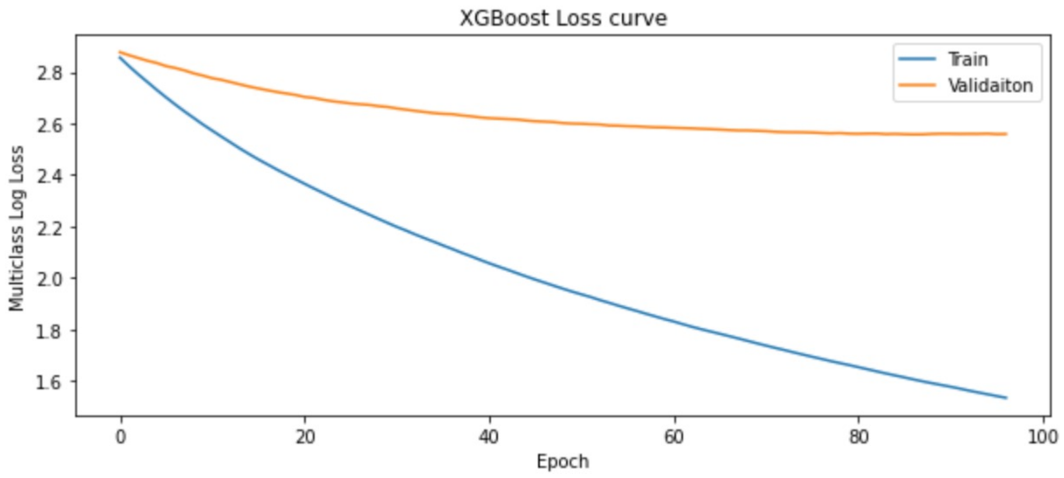
Input:

- Number of pixels in each colour

Evaluation:

- Preprocessing worse than hand-picked colours

Model	Accuracy	F1-score
Human average	54.6 %	52.5 %
XGBoost on stats	17.0 %	13.9 %
XGBoost on extracted base-colours	17.0 %	13.7 %
XGBoost on hand-picked colours	26.8 %	25.0 %
XGBoost on type-colours	23.2 %	19.8 %



Predicting the type

Model	Accuracy	F1-score
Human average	54.6 %	52.5 %
XGBoost on stats	17.0 %	13.9 %
XGBoost on extracted base-colours	17.0 %	13.7 %
XGBoost on hand-picked colours	26.8 %	25.0 %
XGBoost on type-colours	23.2 %	19.8 %

Model type:

- XGBoost Classifier

Preprocessing:

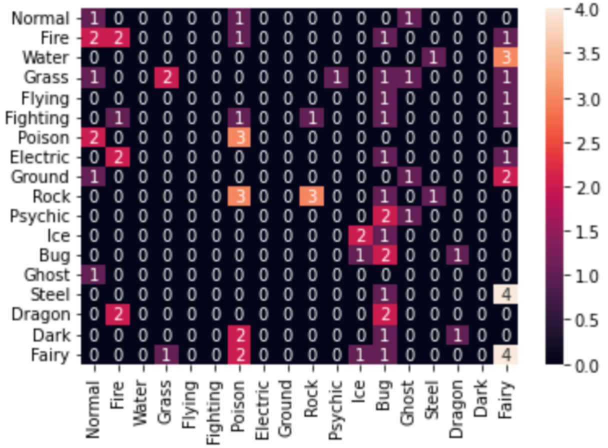
- Classified each pixel as a colour, based on proximity to list of RGB colours

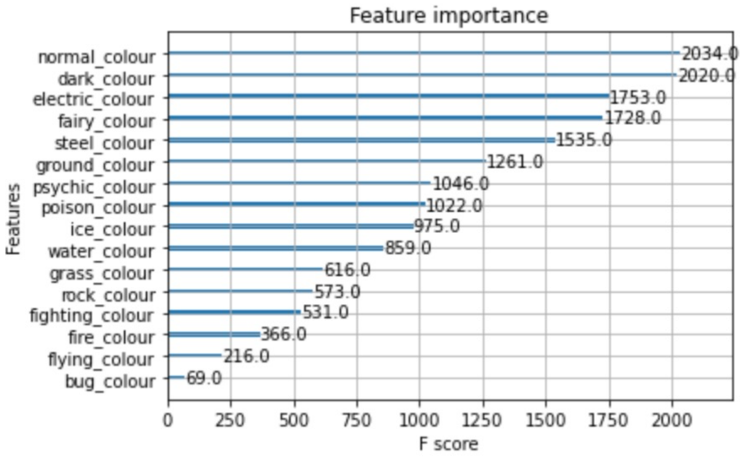
Input:

- Number of pixels in each colour

Evaluation:

- Preprocessing worse than hand-picked colours





CNN

PyTorch

- Number of epochs: 20
- Learning rate: 0.001
- Momentum: 0.9
- Pretrained network: ResNet-18

- Also tried TensorFlow, however PyTorch proved easier to use.
- Tried three other pretrained networks, but ResNet-18 had best performance

CNN - SHAP

PyTorch and TensorFlow

- Maximum number of evaluations: 5000
- shap.explainer()

- Some inconsistencies in SHAP predictions compared to the CNN model's predictions

Predicting the type

Model type:

- Modified ResNet18

Input:

- Only images

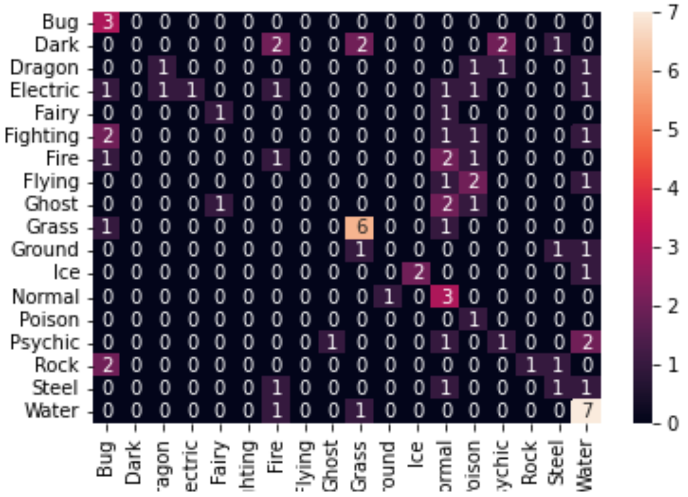
Preprocessing:

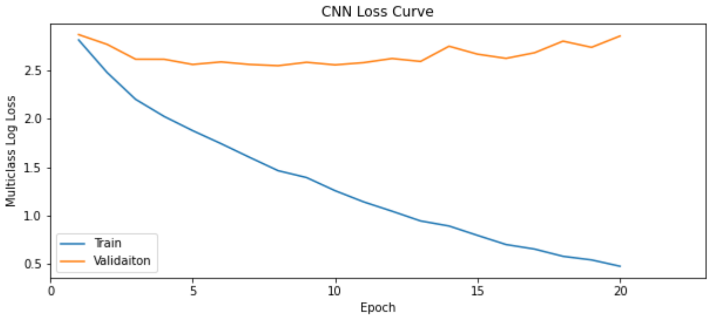
- Normalize pixel values
- Augment train images

Evaluation:

- Very good model

Model	Accuracy
Human average	54.6 %
CNN from scratch	15.9 %
CNN feature extract ResNet18	25.6 %
CNN full ResNet18	28.1 %
CNN full ResNet18 + augment	34.2 %





Clustering

UMAP – nearest neighbors

- Number of neighbors in cluster: 50
- Minimum distance between points: 0.0
- Number of components: 2
- Number of epochs: 100
- Metric (loss function): "minkowski"

Auto-encoder

Keras variational auto-encoder

- Model structure: 87 M parameters
- Encoder: 4 convolutional layers, 3 dense layers
- Latent space (2, 3, 4, 128, 256, 512 dim)
- Decoder: 1 dense layer, 5 convolutional layers
- Learning rate: 0.0004
- Number of epochs: 100
- Optimizer: Adam

Last true-model

- If presented the true type of the previous Pokémon, one might simply guess the same type for the next Pokémon – following the logic of evolutions and some system in the "Pokédex" order
- The predictions of a such models have an accuracy of 45.12 % for Generation 8 (Test set)

- Fault in comparability with other models: Last true-model is presented for truth values for the test set.

Ability frequency-model

- Based on a histogram of the most frequent type for each individual ability, a model may classify each Pokémon to the type which is most common for its specific ability.
- Essentially a tree-based model with a branch for each ability as only divisor.
- Works very well for generations 1-7, but fails predicting generation 8 due to the introduction of many new abilities.
- Comparing same set of abilities yields a

Accuracy, train: gen1-7, test: gen1-7	67.57%
Accuracy, train:gen1-7, test: gen8	25.90%
Accuracy, train:gen1-7, test: gen8 excluding new abilities	46.74%
No. abilities in gen1-7	87
No. new abilities in gen8	45
No. abilities in gen1-8	132