Machine Learning for Physicists and Astronomers: Course Project

Stockholm University, Spring Term 2024
Max Maschke

Mar 08 2024

Contents

1 Introduction

This project report was created as mandatory course work for the lecture Machine Learning for Physicists and Astronomers given at Stockholm University in the spring of 2024. The task was to choose one lower and one higher complexity task from a number of presented real-world datasets and attempt to model them using different machine learning approaches.

For the lower complexity task, we (i.e., I) chose the superconductor dataset due to a personal interest in solid state physics. For the higher complexity task, we went with galaxy classification as to feature one regression and one classification task in this report.

The approaches taken and results obtained are presented and discussed hereafter.

2 The SuperCon Dataset

2.1 Superconductivity

Superconductivity is the phenomenon of vanishing electrical resistance or, equivalently, infinite electrical conductivity in certain solids below a critical temperature T_c. This is surprising because, classically, one would expect a non-zero resistance at $T = 0$ caused by charge carriers scattering off of lattice defects. Microscopically, superconductivity was first explained by Bardeen, Cooper and Schrieffer in what is now known as BCS-theory. The relevant mechanism that causes superconductivity is the formation of charge carrier pairs with equal and opposite momentum and spin, mediated by a tiny inter-electron attraction caused by fluctuations in polarisation due to lattice vibrations, i.e. phonons. Since these so called Cooper pairs have zero total spin, they behave like bosons and can form a Bose-Einstein-condensate which can carry a current without dissipation [1].

Superconductors are interesting for practical application because, to just quote one of many examples, their large scale use in power grids could eliminate a large part of the losses incurred by transmitting electrical currents over large distances. However, the low transition temperatures of most materials identified as superconductors so far inhibit their use to this end because the effort needed to cool the metals below their critical temperature far outweighs any prospective gain in efficiency. For this and other reasons, the search for high-temperature superconductors has been one of the most active areas of research in physics for decades.

As there is to this day no complete theory of high-temperature superconductivity, it is exceptionally difficult to predict the critical temperature of a material from its basic, defining properties. In this section of the report, we will explore and evaluate a simple data-driven approach to modelling T_c from a small number of material features.

2.2 Properties of the Dataset

The data supplied for this task is published at [2]. The author claims that it is an excerpt of the SuperCon dataset, but the provided URL is dead since the SuperCon dataset has been moved to a new website since [2] was originally published. This immediately makes interpretation of the dataset slightly difficult, because no extensive information on the nature of the features of the entries of [2] are provided, and they do not match with the features present in the SuperCon dataset that can now be found at [3], as the publishers of [3] claim that their data is a re-edited version of the original dataset, which is now no longer available online.

Due to this fact, some educated guesses had to be made in order to find out what some of the entries of [2] that will be used for modelling actually are. This is unfortunate but there is no obvious solution to the problem.

The dataset contains basic information on 21263 superconductors as well as each material's critical temperature. It seems likely but cannot be confirmed as discussed above that the features attributed to each superconductor were derived from their chemical composition. Specifically, it seems that for each of the features listed in table 1, the features for each constituent element of the molecule were averaged in different ways. Standard deviations for the different averages are also provided. Additionally, a quantity referred to as "entropy" associated with each feature is present in the data, though its meaning eludes the author.

Table 1: Material features provided as averages and deviations in the dataset [2]

Feature name	Interpretation	Likely units
`number_of_elements`	Number of distinct elements in compound	n.a.
`atomic_mass`	Atomic mass	atomic mass unit (u)
`fie`	Flow-injection extraction*	seconds (s)
`atomic_radius`	Atomic radius	picometers (pm)
`Density`	Bulk mass density	kg/m^3
`ElectronAffinity`	Electron affinity	kJ/mol
`FusionHeat`	Latent heat of fusion	kJ/mol
`ThermalConductivity`	Bulk thermal conductivity	W/(m K)
`Valence`	Number of valence electrons	n.a.

*: Low confidence in correct identification of feature.

All in all, for each of the features, the following statistical quantities are provided:

- `mean`: Likely the arithmetic mean
- `wtd_mean`: Likely a weighted arithmetic mean with unknown weights
- `gmean`: Likely the geometric mean
- `wtd_gmean`: Likely a weighted geometric mean with unknown weights
- `entropy`: Unclear
- `wtd_entropy`: Unclear
- `range`: Likely the difference between the maximum and minimum value
- `wtd_range`: Unclear
- `std`: Likely the standard deviation
- `wtd_std`: Likely the standard deviation with respect to the weighted mean above

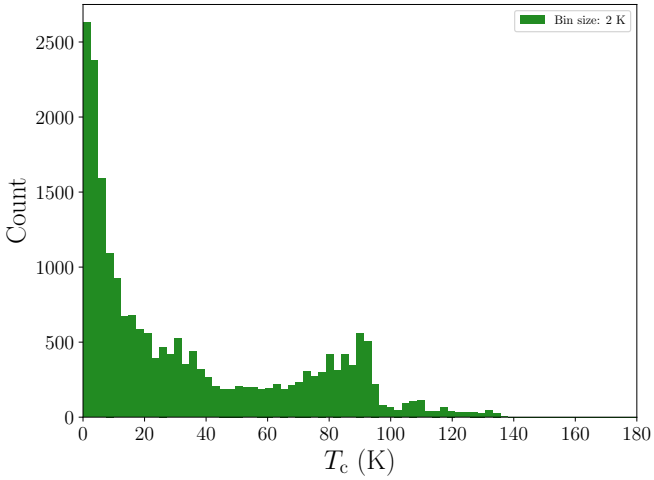


Figure 1: Histogram of critical temperature in the entire dataset. Low T_c materials are by far the most common and there seems to be a strong cut-off at around 90-100 K. This is in line with the physical intuition that high-temperature superconductivity is rare and requires fine tuning of materials to achieve.

Unfortunately, no information on units is available, though an attempt at reconstructing them has been made by comparing values for elementary superconductors to literature.

Due to the fact that the way in which the weighted means were obtained is not known, they shall not be used for further analysis here. The same applies to "entropy", which was discarded from here on. Since the geometric mean and arithmetic mean are highly correlated, solely arithmetic mean and standard deviation of each feature were selected, so a total of 17 features are used for modelling (N.B.: `number_of_elements` is not an averaged quantity and thus has no associated standard deviation).

Before designing a model, it is prudent to visually inspect the data. A simple histogram of the critical temperature, see figure 1, shows that low T_c materials are by far the most common in the dataset, and there seems to be a strong cut-off at around 90-100 K. This is in line with the physical intuition that high-temperature superconductivity is rare and requires fine tuning of materials to achieve. A plot of each of the remaining features against the target quantity, T_c, is shown in figure 2.

Observing each feature's correlation with T_c, it can be seen that for almost all of them, there appears to be a range in which high-T_c materials are especially prevalent. This inspires some confidence that a model trained on the dataset might in principle be able to determine if a given material lies in this range for each feature and might thus show a high critical temperature in experiment. However, it must be stressed here already that superconductivity is a collective phenomenon that involves the entire electronic and lattice structure of a solid. Since the features used here are averaged bulk quantities that are not informed about the manifold and subtle ways that each constituent in a crystal can affect its properties, one would naturally expect some biases in the data, caused in part by how researchers have historically selected materials for investigation.

As just one example, figure 2 seems to imply that a high number of distinct elements in

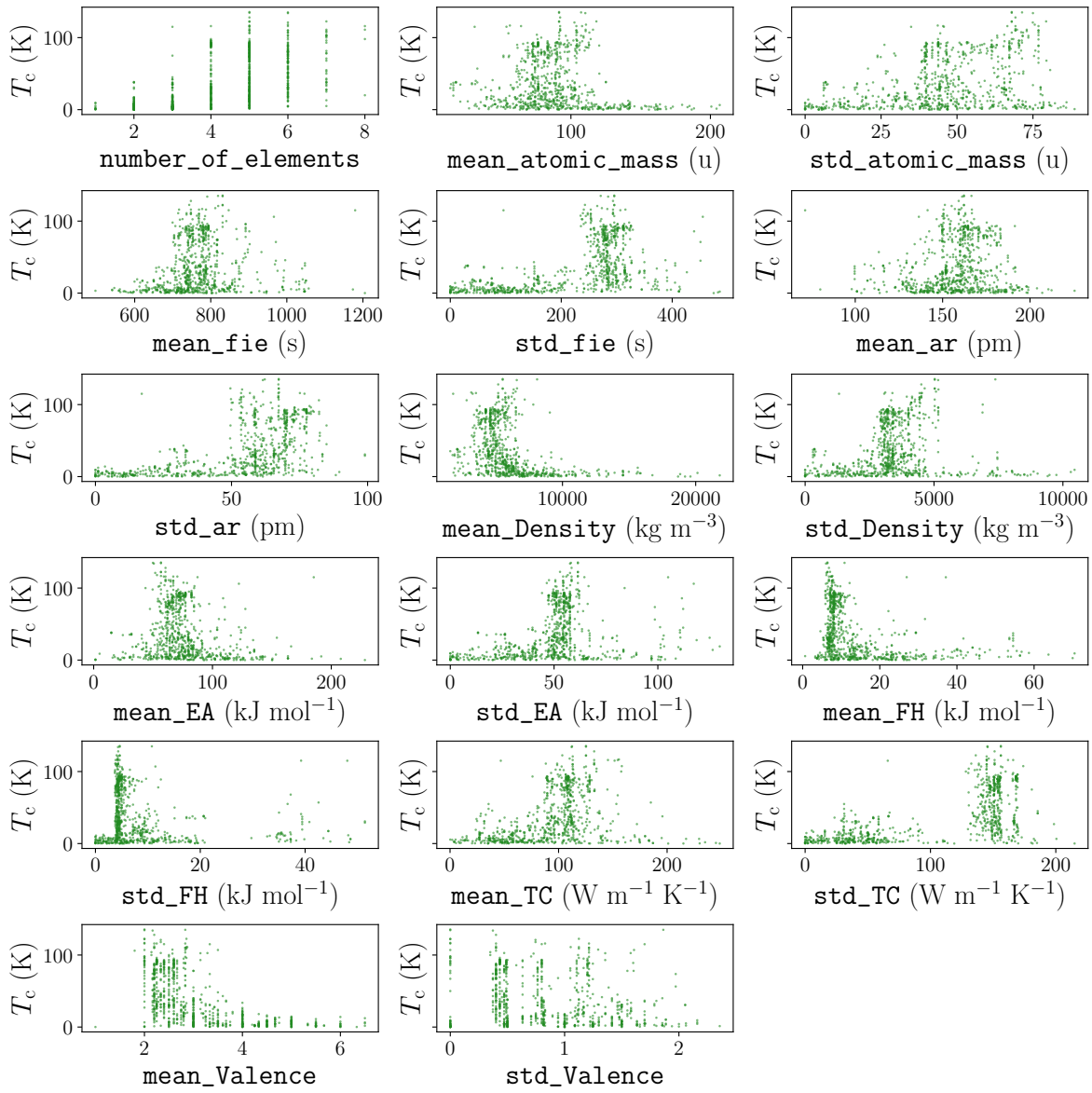


Figure 2: Scatter plot of each of the selected features for modelling versus critical temperature for 800 randomly sampled entries of the dataset. Atomic radius has been abbreviated as `ar`, electron affinity as `EA`, fusion heat as `FH` and thermal conductivity as `TC`. Note that for the quantitiy `fie`, the choice of seconds as the unit is a low confidence guess on part of the author. For almost all of the features, there appears to be a range in which high-T_c materials are especially prevalent, which is confidence inspiring for modelling with the aim of predicting high T_c especially.

compounds seems to correlate somewhat with a higher critical temperature. However, it is obvious that, firstly, there is simply a larger number of distinct compounds made up of a larger number of constituents. Secondly, since each constituent can be thought of as a parameter in a complex optimisation problem, one might expect materials with more constituents to be able to be better fine tuned for some of the properties that incur a higher T_c. This does not, however, imply that the simple fact that a given compound is made up of many different elements somehow implies a high T_c.

In spite of these difficulties, we will explore the possibility of using the dataset for training T_c predictors in the following section.

2.3 Modelling the Data

Two different modelling approaches shall be investigated and compared as to their success. The first is a simple linear regression in all of the features. The second are deep neural networks, which offer a way larger number of parameters to be tuned but are more difficult to interpret.

For both approaches, the features are first scaled using the `StandardScaler` method from `scikit-learn` [4], which linearly scales the feature to have zero mean and unit variance. This has two advantages: first, it can prevent problems with gradient estimation for the training of the deep network. Second, it makes it possible to compare feature importance measures for each model respectively.

For both models, the data was divided into a training and a test set, with 70:30 split being used.

2.3.1 Linear Regression

For the linear model, ordinary least squares optimisation is employed, i.e. from N samples (x_i, y_i) we want to find a coefficient vector $\mathbf{w}$ and an intercept b such that the predictor

$$\tilde{y} = \mathbf{w}^T \mathbf{x} + b \tag{1}$$

minimises the mean squared residual S given by

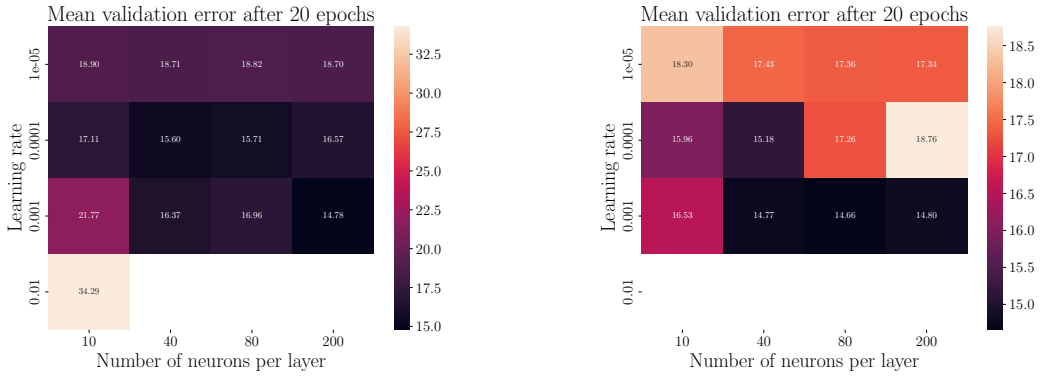
$$S = \frac{1}{N} \sum_{i=1}^{N} (\tilde{y}_i - y_i)^2 \ . \tag{2}$$

This problem has an exact solution that is implemented in `scikit-learn`'s `LinearRegression` method and involves numerically solving a system of linear equations.

2.3.2 Deep Neural Networks

The second modelling approach that is to be explored here is a fully connected feed-forward neural network, or deep neural network. We shall omit a description of what these networks are in a formal sense and how they can be trained here and instead refer to the lecture material for more information.

Unlike the simple linear model described above, a neural network can be constructed in infinitely many different ways. Choosing a model architecture that works well for a given dataset is a non-trivial task. Here, two ways of finding suitable sets of hyperparameters were explored: a simple grid search as well as the `KerasTuner` package [5].



(a) Two hidden layers

(b) Three hidden layers

Figure 3: Square root of mean squared error of neural networks with different numbers of neurons per layer n and learning rates η for two 3a and three 3b hidden layers after 20 epochs of training. White squares indicate that loss diverged during training. Note that the colour bars in both plots have different ranges.

The model was implemented using the Keras API [6] for TensorFlow [7]. We decided to use m dense hidden layers with n neurons each, all using the standard rectified linear unit (`ReLU`) activation function. The output layer is a single neuron with a linear activation function. Also using a `ReLU` in the output neuron would enforce $T_\mathrm{c} > 0$, however, the results show that this was not needed.

Stochastic gradient descent (SGD) was used to train the model, optimising the mean squared error. In training, 20% of the training set was used as validation data to monitor the training process for overfitting. The learning rate η was chosen as the third hyperparameter that was to be determined for optimal performance.

For the grid search, we explored the parameter space spanned by $n \in \{10, 40, 80, 200\}$ and $\eta \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ for networks with $m = 2, 3$ hidden layers. We trained the models for 20 epochs each, with a batch size of 32. The square root of the loss, which we define to be ΔT, is shown for all models tested in figure 3. White squares indicate that the loss diverged during training. It can be seen that $\eta = 0.01$ seems to be a particularly bad choice for the learning rate since it shows divergent behaviour during training. The optimal set of parameters indicated by this grid search is $(m, n, \eta) = (3, 80, 0.001)$.

The `KerasTuner` package provides a more convenient way of implementing hyperparameter optimisation than manually implementing grid search. Furthermore, it implements statistical methods that aim to more efficiently search the parameter space. We chose to use the `BayesianOptimization` function and searched again for an omptimal set of (m, n, η). The optimal model found by the package had parameters $(m, n, \eta) = (3, 125, 0.00109)$, which is not too dissimilar from the set of parameters found by our more pedestrian grid search.

We then chose to train this model for 200 epochs with a batch size of 16. The training and validation loss curves during the training process can be seen in 4. The validation loss shows large fluctuations initially, which die down towards the end of the training process, which indicates that the ability of the model to generalise to unseen data improved.

The results of applying both the linear model and the neural network to the test set are discussed int he following section.

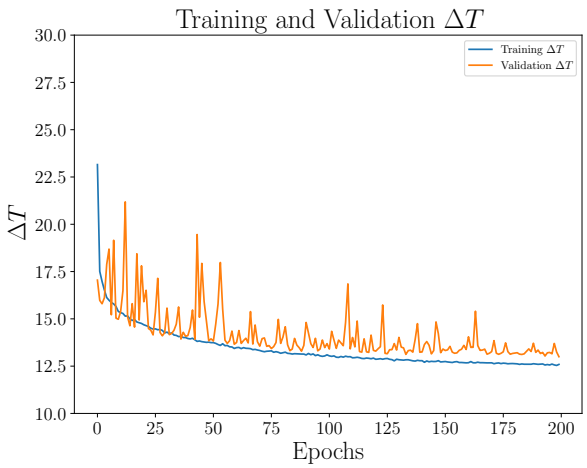


Figure 4: Evolution of the square root of training and validation loss over 200 training epochs. The slightly erratic behaviour of the validation error calms down towards the end of the training, which implies that the model generalises increasingly well with further training.

2.3.3 Results

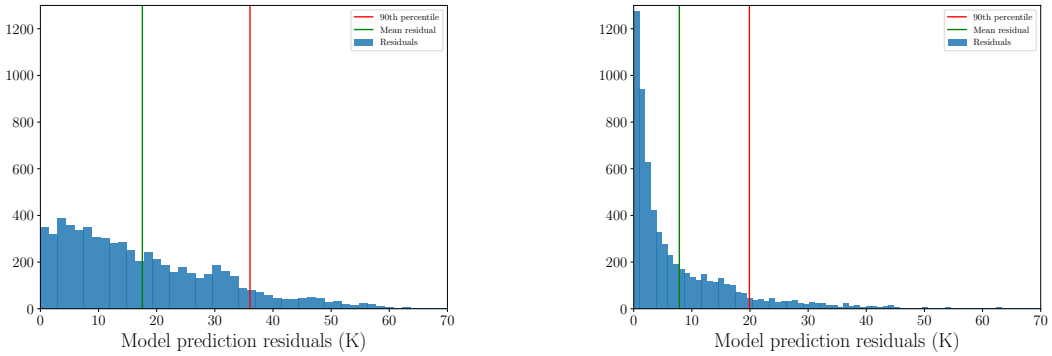
First, we present the prediction errors of both models as histograms in figure 5. It can be seen that high errors are more prevalent in the linear models predictions and that they appear to fall of linearly over a large domain. On the contrary, the deep network shows a super-linear drop in occurrence rate towards higher errors, as the average error is on the order of 9 K and 90% of predictions incur errors less than 20 K, while the linear model almost reaches such high errors on average.

This observation alone should be enough to choose the network model over the linear model, and this conclusion is further supported by figure 6 which shows predicted versus true critical temperatures. An ideal predictor ought to produce as little deviation from the axis bisection as possible.

Figure 6 includes not only the performance on the test set but also the training set, which is useful to see if the model generalises well onto unknown data.

A major flaw of the linear model is its failure to predict critical temperatures over 80 K, which the network does not struggle to do. Recall that around the 80 K range, higher T_c become rare in the dataset which might explain the behaviour of the linear regression. Additionally, the linear model often predicts nonphysical negative T_cs while the network barely does so at all. For true T_c between 0 and 100 K, both models seem to severely underestimate T_c more often than severely overestimate it. In the $T_c > 100$ K regime, the network seems to perform well despite a certain statistical error, which is encouraging for applying it to high-T_c material search. The logarithmic plots 6c, 6d show model performance at low T, where the neural network also vastly outperforms the linear model.

The best neural network trained yields an R^2-score of $R^2 = 0.86$. A 2018 study [8] that used a similar approach but more involved material features such as information on crystal lattice symmetries and orbital configuration of constituents achieved an $R^2 = 0.88$, which implies that the additional effort undertaken in that paper yields only a slight predictive advantage



(a) Linear model

(b) Neural network

Figure 5: Frequency of test set model prediction errors for the linear 5a and deep model 5b. The residual frequency seems to fall off linearly for the linear model while it clearly falls off superlinearly for the neural network. For the network, 90% of predictions incur an error of less than 20 K, while for the linear model the 90th percentile is roughly 36 K.

and the raw elementary properties used here are competitive as features for T_c prediction. This is surprising but also encouraging because they are much more attainable, as they do not require possibly computationally expensive methods to be applied to every candidate material to extract features.

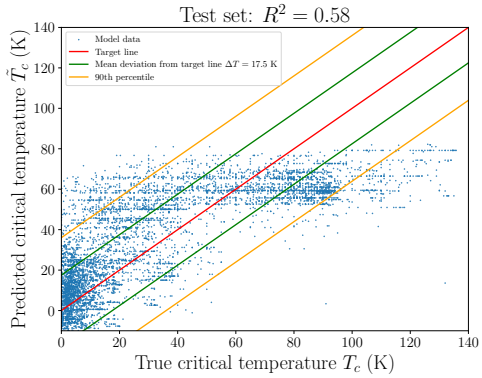
Explaining the influence of each feature on the network is well known for being a difficult problem. For the linear model, one could look at the coefficient for each feature for insights, however, we chose against this because this method is not available for the network.

Instead, we chose to calculate the Shapely values for both models. Shapely values are a concept from game theory that aim to quantify each players contribution to success in a cooperative game by considering the chance of success if they did not participate. This can be applied to neural networks by thinking of the features in the input layer as the players and the regression or classification task as the game. Shapely values for neural networks are usually estimated using various methods because an exact calculation would require to re-train the network for all combinations of features from the full feature set.

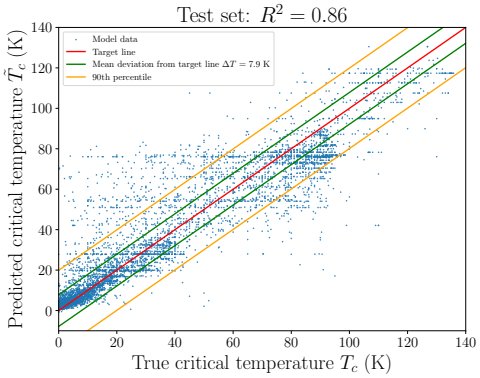
We use the `shap`-package for python [9] to obtain the plot shown in figure 7. The features are sorted in descending order of mean absolute value of the Shapely values from the top, i.e. for the network,`std_atomic_radius` has the highest mean absolute value shapely value etc. The inclusion of colour information shows how a high or low value in the feature impacts the model output. Put simply, if the point cloud on the right hand side of the x-axis is red, larger feature value implies higher T_c and vice versa.

For example, for the linear model, a high average density seems to be strongly associated with a low T_c, while the variance in density seems to have minimal impact on the model prediction. This is consistent with the distribution of the data seen earlier in figure 2.

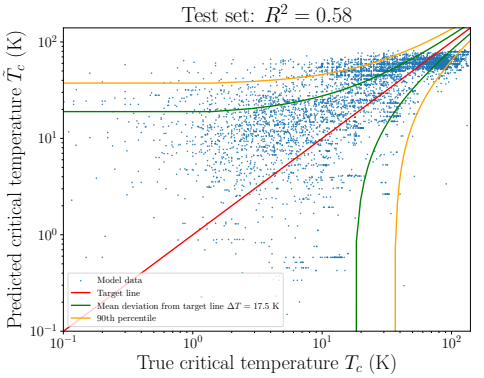
One simple sanity check to perform having calculated these metrics is to see if there are any features which scale positively with T_c for one and negatively for the other model. Encouragingly, this cannot be observed with one notable exception, `std_thermal_conductivity`, which shows a completely different qualitative distribution of Shapely values for both models. How-



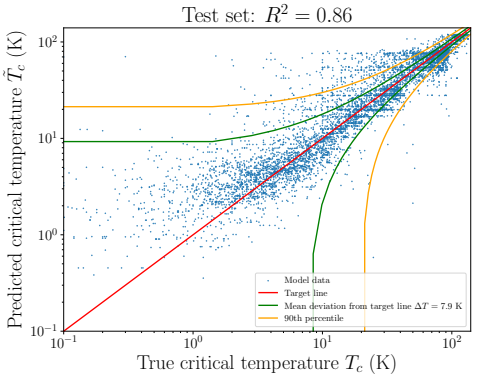
(a) Linear model, test set



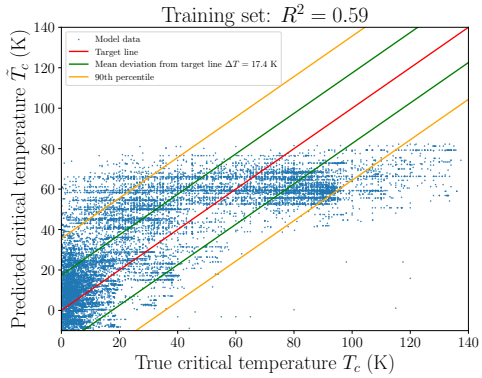
(b) Neural network, test set



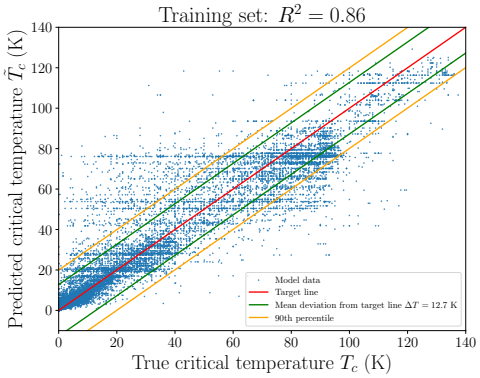
(c) Linear model, test set, log axes



(d) Neural network, test set, log axes

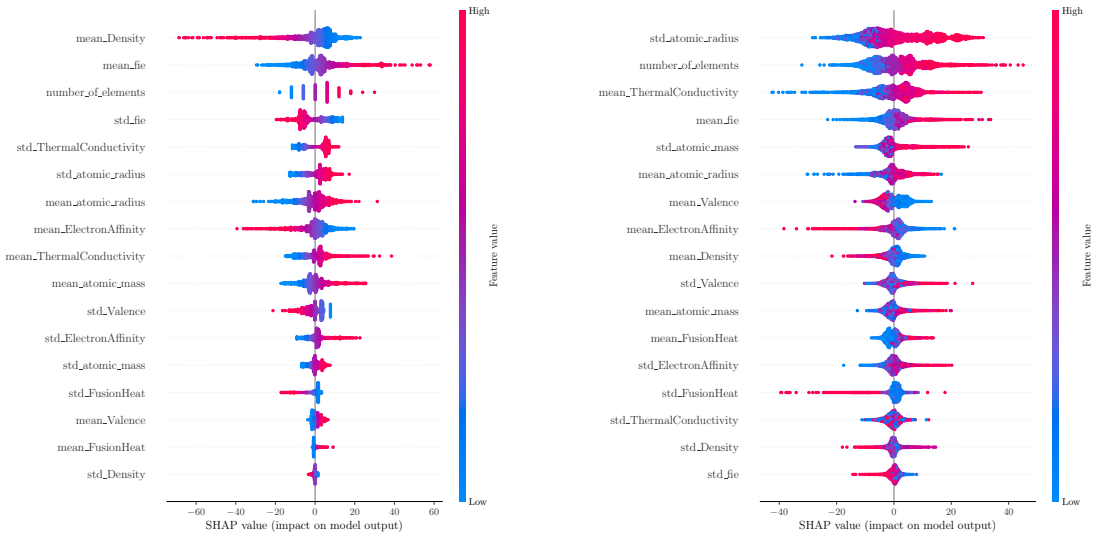


(e) Linear model, training set



(f) Neural network, training set

Figure 6: Predictive performance of the linear and deep models for the test set and the training set. Training set performance in 6e, 6f is included to show that both models generalise well. A major flaw of the linear model 6a is that it fails to predict critical temperatures over 80 K, whereas the neural network fares much better in this regard. Additionally, the linear model 6a often predicts nonphysical negative T_cs while the network 6b barely does so at all. For true T_c between 0 and 100 K, both models seem to severely underestimate T_c more often than severely overestimate it. In the $T_c > 100$ K regime, the network seems to perform well despite a certain statistical error, which is encouraging for applying it to high-T_c material search. The logarithmic plots 6c, 6d show model performance at low T, where the neural network also vastly outperforms the linear model.



(a) Linear model (b) Neural network

Figure 7: Estimated shapely value distribution for the linear 7a and 7b network model. Features are ordered from top to bottom according to importance, i.e. average absolute value of the Shapely value.

ever, which feature has the largest impact on the model output is not consistent between both models, but this is not necessarily surprising due to the vastly different model architectures. One would expect a neural network to also be able to better extract information from features that do not have a large impact on the linear model. For example, `mean_Valence` is one of the least important features for the linear model but one of the most predictive ones for the neural network.

Both models are highly influenced by the number of elements in a compound. We already discussed above why this is to be expected but does not necessarily hint at a physical mechanism that connects a large set of elements with high T_c per se.

It is interesting to observe that both models take the mean thermal conductivity into account strongly. One of the main mechanisms of heat transfer in crystals is via phonons. The coupling of lattice electrons to phonons is also a prerequisite for superconductivity. It could thus be worth investigating whether this is not just a correlation but actually hints at some sort of causal connection.

Further, we note that materials in which the constituents show on average large electron affinities tend not to show high temperature superconductivity. This could be due to electrons being too localised to overlap sufficiently to form pairs at higher temperatures.

It should be noted that different algorithms are used by the `shap`-package for estimating the Shapely values for different model architectures so comparisons should take this into account.

Another problem that was not previously discussed is that the dataset only contains superconductors. However, not all materials show superconductivity and this is not reflected in the dataset. The models may thus predict a critical temperature for materials that do not show this phase transition at all.

The dataset also does not include information on the cooper pair formation mechanism, which is not identical for all superconductors. Incorporating this information if available for a

previously untested material might lead to better performance.

To conclude, we have achieved a predictive performance competitive with recent literature using a neural network. Observing the feature importance hints at two possible connections between bulk constituent features and critical temperature.

3 The Galaxy Zoo Dataset

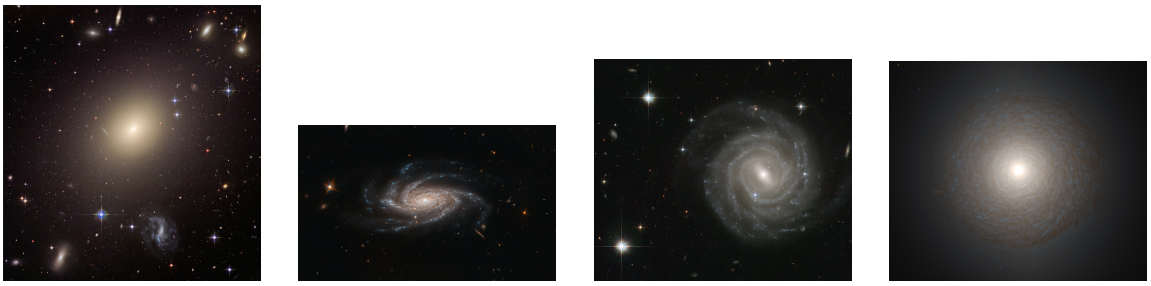
3.1 Classification of Galaxies

Before the advent of high-magnification telescopy, humanity's knowledge about the different objects on the night sky was inherently limited and tainted by superstition. With Galileo's telescope, researchers first realised that the band visible to the naked eye known as the milky way is a structure made up of many stars. Figuring out what this structure looked like from the outside turned out to be a difficult problem, given that we're located inside it. This was partly because other galaxies were difficult to observe and even more difficult to identify as objects that do not belong to our own galaxy.

Table 2: Galaxy classification system introduced by Hubble in 1926 [10]. The n in the elliptical galaxy symbol refers to the leading decimal in the galaxy's ellipticity.

Type	Subtype	Symbol
Elliptical		En
Lenticular		S0
Spiral		S
	Early spirals	Sa
	Intermediate spirals	Sb
	Late spirals	Sc
	Barred Spirals	SB
	Early barred	SBa
	Intermediate barred	SBb
	Late barred	SBc
Irregular		Irr
Faint		Q

In fact, that some nebulae are actually separate objects was only proven by Hubble in 1923 [11]. It was also Hubble who first tried to devise a system for classifying galaxies, which were being discovered at an accelerating rate. He proposed the categories listed in table 2 in 1926 [10]. Broadly speaking, he suggested sorting galaxies into ellipticals, spirals, spirals with a central bar and irregulars, as illustrated in figure 8. The choice of words of "early" and "late"



(a) Elliptical Galaxy ESO 325-G004 (b) Spiral Galaxy NGC 2008 (c) Barred spiral Galaxy UGC 12158 (d) Lenticular Galaxy NGC 1387

Figure 8: Examples of the major galaxy classes introduced by Hubble. All ©NASA.

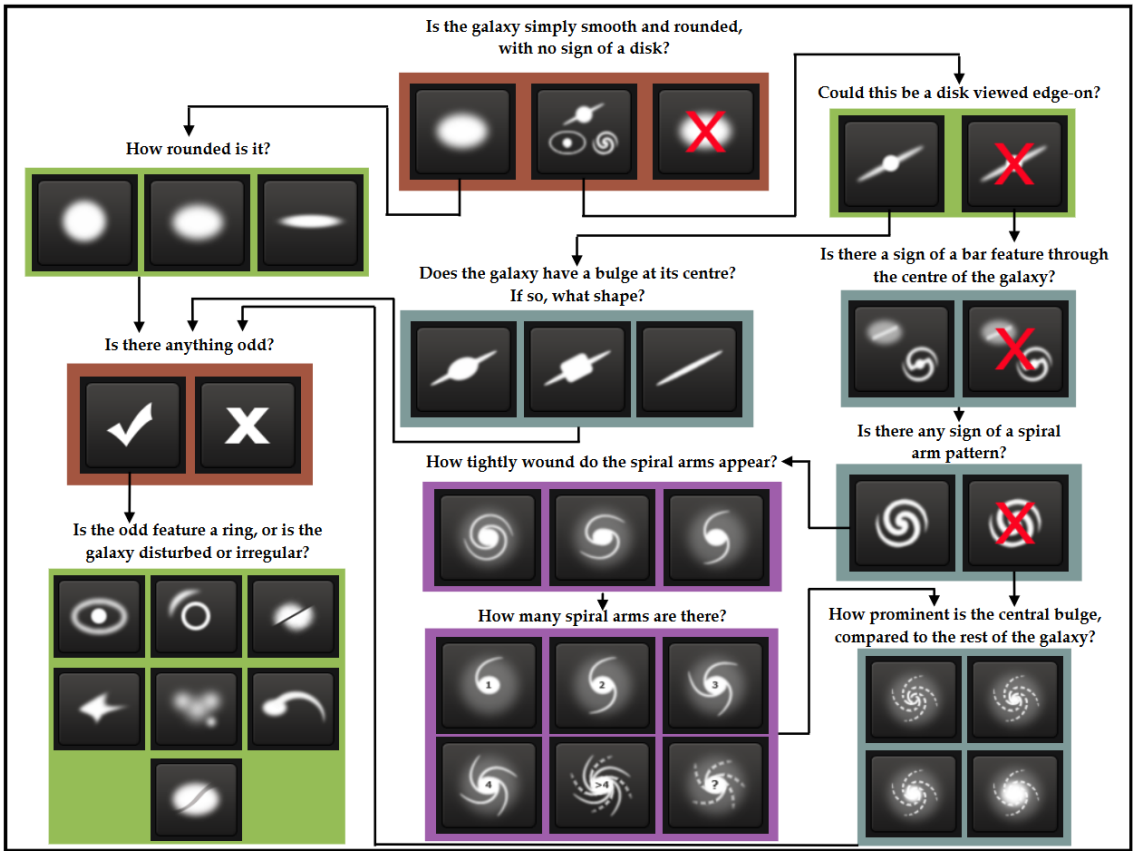


Figure 9: Question tree of the GalaxyZoo 2 survey [12].

spiral galaxies for what is, on a morphological level, actually referring to the tightness of the spiral pattern, does not carry however any meaning in a temporal sense, i.e. referring to the galaxy's evolution [10]. In what have almost been 100 years since, there have naturally been multiple attempts at extending Hubble's system. Nonetheless, it is still commonly used.

The large amount of galaxies that are catalogued today call for automatic classification based on astrophotographical data. This in turn makes it a prime candidate for applying supervised learning techniques, due to the large number of pre-labelled information already available.

One effort undertaken to increase the availability of labelled data is the GalaxyZoo citizen science project where participants were asked to answer a series of questions about galaxies shown to them on a volunteer basis. It is the second iteration of this dataset that is to be looked at in further detail here, GalaxyZoo 2 [12].

3.2 Properties of the Dataset

The version of the dataset used in this project was not directly published by GalaxyZoo but was rather posed as a challenge to users on the machine learning and data science website Kaggle in 2013 [13]. Users were invited to model a subset of the original GalaxyZoo 2 survey. While the original dataset contained 304122 galaxies, the Kaggle dataset has been shrunk to 61578 galaxies to be used as training data and a further 79975 galaxies to be used as test data.

Due to the nature of the Kaggle challenge having been a competition where prize money

was involved, the test images were not labelled and can thus not be used for the remainder of this analysis.

The remaining galaxy data comes in pairs of one 424×424 pixel jpg-images and one 37 column label vector each. Galaxies are also given a unique identification number. The GalaxyZoo 2 survey did not just aim to sort galaxies into Hubble's classical categories but to gather more detailed information on galaxy morphology. Volunteers answered questions along a question tree and it is the averaged share of users' answers to these questions that are included as label data in the dataset. The full question tree is shown in figure 9.

There is thus no hard labelling information available like for a classical image classification task. Technically, the data does not even directly include a set of labels with probabilities assigned to them at all because the entries of the label vector follow the question tree. The original Kaggle challenge asked participants to fully model the choice probabilities for all questions along the question tree. Here, we want to take a different approach that is easier to interpret[1] because it amounts to a reduction in complexity.

3.2.1 Label choice

After initial inspection of the original dataset and a few failed attempts at modelling it, we decided to simplify the task. This would not only make it more feasible to achieve an interesting result with the available computational resources but also yield itself better to interpretation.

We chose to transform the question tree information into label information by making the following assumptions:

- If the galaxy is smooth according to the first question and has no odd featured according to the second question, it is **elliptical**.
- If the galaxy has features according to the first question, is not edge-on according to the second question, has no bar according to the third question and shows a spiral pattern according to the fourth question, it is **spiral**.
- If the galaxy has features according to the first question, is not edge-on according to the second question, has a bar according to the third question and shows a spiral pattern according to the fourth question, it is a **barred spiral**.
- If the galaxy has features according to the first question and is viewed edge-on according to the second question, it is **edge-on**.
- If the galaxy has features, is not edge-on, has no bar and no spiral, or is not a galaxy according to the first question, we sort it into **other**.

This yields in a five column label vector which we can interpret as a probability distribution for a galaxy in the dataset being in either of the five classes decided upon above.

In practice, we get the probabilities by multiplying the normalised shares of responses for each subsequent question, e.g.

$$P(\text{galaxy} = \text{spiral}) = P(\text{has features}) \cdot P(\text{not edge-on}) \cdot P(\text{no bar}) \cdot P(\text{has spirals})\,. \qquad (3)$$

[1] The winning entry to the Kaggle competition made no attempt at interpreting its results at all - it's all about the money, after all, and apparently not so much about the knowledge.

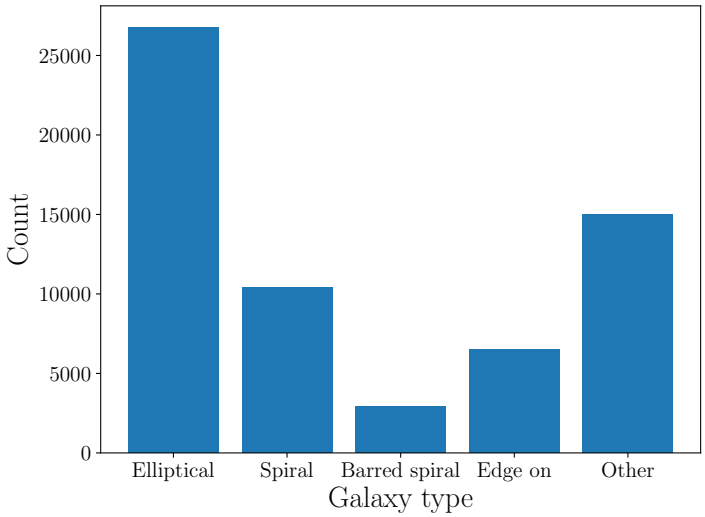


Figure 10: Count of galaxies labelled according to (4) in the dataset. It can be seen that elliptical galaxies seem to outnumber spirals by a factor of ca. 2.

If desired, we can now visualise the distribution of the labels within the dataset. This comes with the caveat of making the assumption that the class label L should follow from

$$L = \text{argmax}_{\tilde{L}}\, P(\text{galaxy} = \tilde{L})\,. \tag{4}$$

For cases where the maximum is pronounced, this is not a controversial choice. There are however examples in the dataset where two class probabilities are very close to $1/2$. In these cases, the label information is to be taken with care.

Nonetheless, the distribution of labels can be seen in figure 10. Elliptical galaxies seem to be the most prevalent in the dataset, outnumbering the sum of spirals and barred spirals by a factor of roughly two. The "other"-category is the second most prevalent.

It is also the "other"-category that posed a significant challenge during model training later on, which is why we ended up removing it from the set of galaxies considered for modelling. Poor performance of modelling this category likely stemmed from the fact that it encompassed too many different types of galaxies, many of which are also similar to the other classes which made distinction difficult for the model. We elaborate on this problem in a later section.

3.3 Modelling the Data using Convolutional Neural Networks

The most common machine learning model architecture for image classification are convolutional neural networks. We refer once again to the lecture material for a description of their operation.

We once again use the `Keras`-API for `TensorFlow` to implement, train and evaluate the network.

We employ a 70:30 train-test split of the dataset.

3.3.1 Data augmentation and class weighting

The original images provided have two problems: They are comparatively large and there are too few of them. The first makes training and evaluating the model more computationally expensive and requires a larger model to capture the image structure. The second problem makes overfitting later on more likely. Luckily, we can alleviate both problems rather easily.

Firstly, we can downsample the images. An ideal downsampling ratio should preserve enough detail as not to inhibit classification by the model, while reducing image size as much as possible. We chose to initially reduce the images to 70×70 pixels. This is done by first center cropping the images to the central 212×212 square, where the galaxies of interest lie in almost all cases, and then downsampling to the desired resolution. The cropping might bring the added benefit of removing possible distractions from the images.

Secondly, since the CNN only features innate translational symmetry, we apply random 90° rotations, transpositions, mirroring and a random crop to 60×60 pixels using the albumentations library. Since a rotated or flipped image is like a new, previously unseen image to the CNN, we expect this to and can observe it to help increase model accuracy after training.

Observing the class prevalence in figure 10, one notes a large imbalance. If trained without any sort of class weighting, the CNN tends to trick us by essentially always guessing the most prevalent category.

We avoid this problem due to our choice of loss function. Categorical crossentropy is a common choice for classification tasks. This loss function is agnostic to the prevalence of labels in the dataset, though, and can thus show the aforementioned issue.

`Keras` implements a variant of the categorical crossentropy called categorical focal crossentropy FL which weights each part of the sum over the probabilities by a factor of α_i:

$$\mathrm{FL} = -\sum_i \alpha_i (1 - \tilde{p}_i)^\gamma \log \tilde{p}_i \tag{5}$$

$$\text{where } \tilde{p}_i = \begin{cases} p_i, & y_i = 1 \\ 1 - p_i, & \text{else} \end{cases} \tag{6}$$

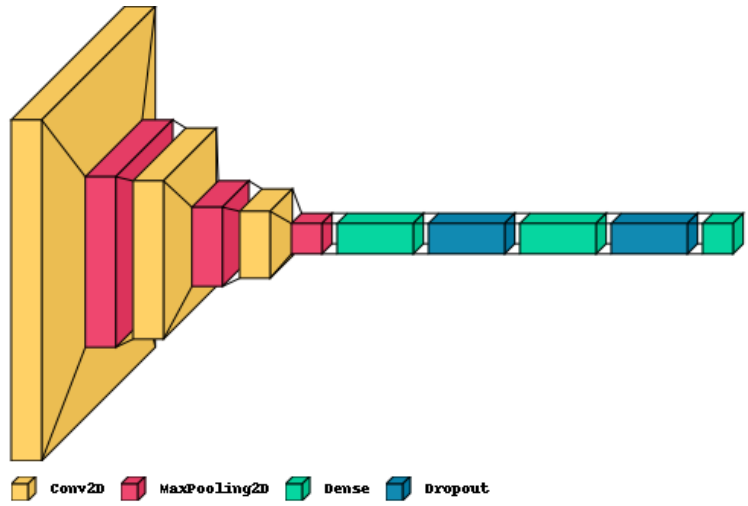
The additional parameter γ can be used to fine-tune the behaviour of the loss function. We left it at $\gamma = 2$, the `Keras` default. The α_i are in turn determined by `sklearn`'s `compute_class_weight` function, as recommended by the `Keras` documentation.

Intuitively, if a class is overrepresented in the data, choosing a small α for that class will equalise its influence on the loss. Conversely, a large α for an underrepresented class will do the same for that class.

3.3.2 Model architecture and hyperparameter selection

As mentioned before, we employ a convolutional neural network (CNN) to tackle our classification task. We decide on the following general architecture:

- Three `ReLU`-convolutional layers with kernel sizes of 5×5, 3×3 and 3×3 in that order with max-pooling of kernel size 2×2 after each.
- l `ReLU` dense layers with n neurons.
- A dropout layer after every dense layer with dropout probability $p = 1/2$.



- First conv. layer: 50 5×5 filters
- Second conv. layer: 100 3×3 filters
- Third conv. layer: 200 3×3 filters
- Two dense layers with 500 neurons each
- Dropout after each dense layer
- Learning rate $\eta = 5 \cdot 10^{-4}$

Figure 11: Model architecture for galaxy image classification informed by hyperparameter space searches. `Keras` model visualised using `visualkeras` [14].

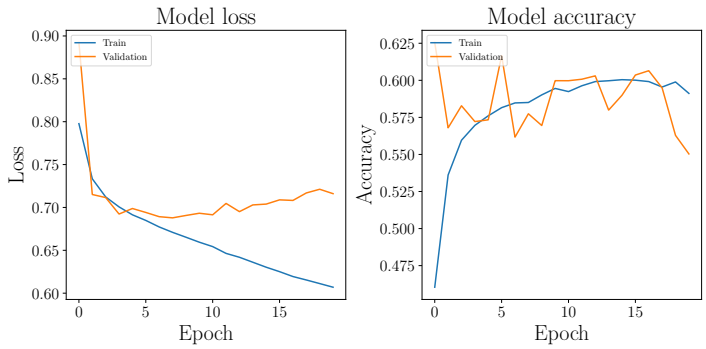
- A `softmax` dense layer with the number of output neurons matching the number of classes.

We decide to optimise our model using the Adam optimiser implemented in `Keras`, which leaves us with the following set of 7 model hyperparameters:

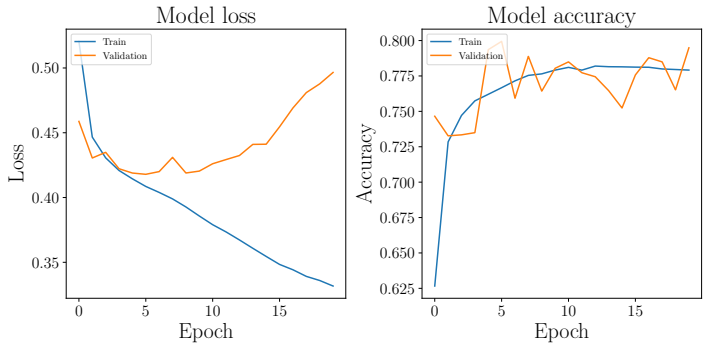
- The number of filters in each of the three convolutional layers, respectively,
- the number of dense layers l,
- the number of neurons per dense layer n,
- the learning rate η and
- whether to use dropout after the dense layers or not.

We again use the `KerasTuner` package's `BayesianOptimization` routine to find an optimal set of hyperparameters. We summarise our choice informed by the parameter search in figure 11. Consequently, we train our model using the augmented dataset and all five galaxy classes we devised for 20 epochs, using 20% of the training set as validation data using training. The evolution of training and validation loss and accuracy can be seen in figure 12a. The achieved categorical accuracy of around 60% is not satisfactory, so we look at the confusion matrix in 13a to see where the problem lies. While there seems to be some trouble in telling barred spirals and regular spirals as well as ellipticals from edge-ons, which is intuitively understood, the "other" is misidentified along the board and it seems almost as if the model just guessed there. We think this might be due to the large variance of galaxy (and, rarely, star) morphology in this category which stems from the naive way in which we attributed images to it. We thus decide to remove the "other" galaxies from the dataset henceforth, which naturally reduces the size of the dataset to 186428 images.

Training evolution of a model trained on the modified data can be seen in figure 12b. We evaluate the model using a callback, choosing the iteration that showed the lowest validation



(a) Including "other"



(b) Excluding "other"

Figure 12: Training and validation loss and categorical accuracy for the CNN described in figure 11 over 20 epochs including the "other" category 12a and excluding it 12b. Both models show strong signs of overfitting, which is why callbacks were used to select an optimal model for evaluation. The model from which the "other" class was withheld clearly fares much better in terms of achieved validation accuracy.

loss[2]. Both models clearly overfit, possibly indicating that they might benefit from further regularisation efforts. Alas, none tried yielded noteworthy improvements. It might also be a sign that the model architecture could be simplified further, for which we find further evidence later.

3.3.3 Results

To begin, we look at the out of sample confusion matrices shown in figure 13. The model including the "other"-category achieves a categorical accuracy of 57%, while the model from which it was withheld achieves 80%.

It can be seen that the low accuracy for the model in 13a stems mostly from the "other"-category in which many misclassifications occur. We have already discussed above that this may be because the "other"-category is ill-defined and contains too much variation. For example, we sort an elliptical with a ring and a spiral with a ring into the same category according to figure 9, which is bound to confuse the model.

[2]Choosing according to lowest validation accuracy would have made more sense but was impossible due tue a documented bug in the `Keras` version used.

(a) Including "other":
Test set categorical accuracy: 57%

(b) Excluding "other":
Test set categorical accuracy: 80%

Figure 13: Out of sample confusion matrices for the CNN described in figure 11 in- and excluding the "other" category. In-sample matrices not shown for brevity. In- and out of sample accuracies are very close to each other and confusion matrices are also similar in structure, indicating that both models generalise well.
It can be seen that the low accuracy for the model in 13a stems mostly from the "other"-category in which many misclassifications occur. Other than that, both confusion matrices show a similar structure:
Ellipticals are generally detected well but sometimes confused with edge-ons; this is to be expected because edge-on is not a strictly defined term and a tilted disk might look very similar to a highly elliptic elliptical. Furthermore, true barred spirals are detected well but true spirals are often mistaken for barred spirals, which is also not surprising because a) the pictures are very low resolution and the bar only makes up a small portion of the image and b) this is also a case where the transition between both categories is somewhat diffuse.

Other than that, both confusion matrices show a similar structure. Ellipticals are generally detected well but sometimes confused with edge-ons; this is to be expected because edge-on is not a strictly defined term and a tilted disk might look very similar to a highly elliptic elliptical. Furthermore, true barred spirals are detected well but true spirals are often mistaken for barred spirals, which is also not surprising because a) the pictures are very low resolution and the bar only makes up a small portion of the image and b) this is also a case where the transition between both categories is somewhat diffuse.

One might argue that an accuracy of 80% is not a very impressive result given we only consider four categories, and we're sure that a more sophisticated (not necessarily more complex) model architecture with better data preparation could easily perform a little better. However, we would like to stress that the model struggles in categories where differentiation is also difficult for humans because the categories we sort galaxies into are inherently somewhat diffuse, pictures are low-resolution and galaxies can overlap or just be very close to each other on the night sky.

If one wanted to use our model to sort large amounts of galaxy data, it might thus be prudent

to accept its predictions for high-confidence cases and let humans double check it if confidence is low, i.e. when the probabilities predicted for two classes are very close to each other. This would make most efficient use of limited human capacity for such tasks.

We would like to mention that we tried to fine-tune a pre-trained model to our data, but this yielded accuracies of only circa 60% for the four-class problem even after long training so it was not competetive. This might be because these models are usually pre-trained on images of everyday life on Earth, which rarely includes close-up photographs of galaxies. It might also be because of a bad choice in architecture for the replaced dense part of the network, but we ran out of GPU time on Kaggle and so could not try other setups.

Now that we have trained a model with some success, we would naturally like to understand how it works. To this end, we will look at the top layer activations for some exemplary galaxies and then try to understand what sort of features correlate with high activation in the layers.

In figure 14, we have selected five galaxies from the test set and shown the model's classifications, the "true" labels and what the image looks like after applying the top layer filters, biases and activations.

We observe that the spiral in 14a is misidentified as an elliptical. The spiral in 14b is correctly identified. 14c presents an interesting case because the frame contains two galaxies, one of which is clearly edge-on. The model seems to be confused by this as it predicts a barred spiral. In 14d, we see a spiral that's mistakenly identified as a barred spiral. 14e shows an edge-on that's correctly identified. We further see that for all galaxies, some filters seem to be "dead", i.e. identically zero. This might indicate that model complexity could be reduced without sacrificing performance.

Many of the non-dead-filter filtered images still resemble the original image for all galaxies, while some seem to amplify the background.

All of the misidentifications in figure 14 seem to be explained if one considers our previous remarks when discussing the confusion matrices.

Another approach at trying to understand what the filters have learned is to use gradient ascent to find images that result in maximum mean activation of a filter when applied. To achieve this, we adapt code presented as an example in the `Keras` documentation [15].

The method works by starting with images containing uniform random noise and then calculating the gradient of the mean of the absolute value of a filters activation and then iteratively taking gradient ascent in the direction of maximum mean activation. This way, we find the images in figure 14.

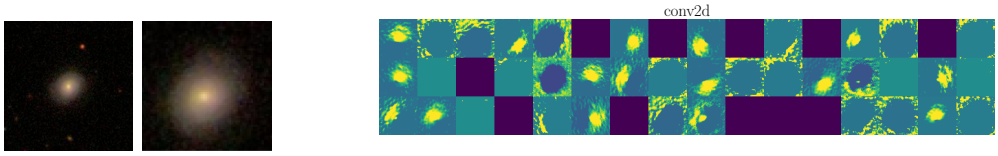
We find that top layer images contain mostly high-level features while low level images are more detailed.

As suspected before when discussing sample galaxy activations, both layers contain dead filters (here: noisy images).

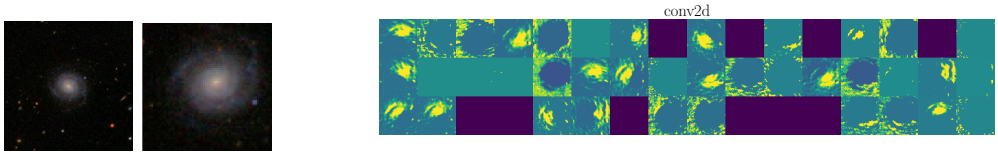
We can identify filters in the top layer that might correspond to detecting bars, spirals, diffuse patterns (ellipticals) and dots. The last layer shows similar behaviour.

Interestingly, the edges of the images in the last layer do not converge away from noise, possibly indicating that the network essentially ignores image edges.

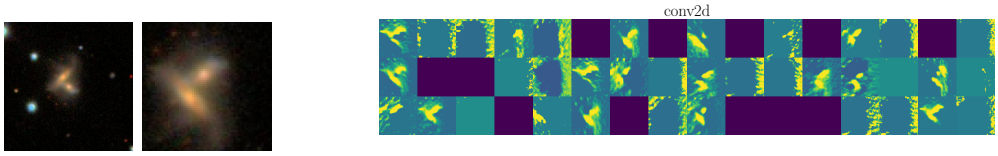
All in all, what we find here is mostly what is to be expected. It would have been interesting to generate images that maximise the output neurons, i.e. for which the model is maximally confident that they belong to a given class. Sadly, we were not successful in applying the above method to the output neurons because the gradient ascent did not converge.



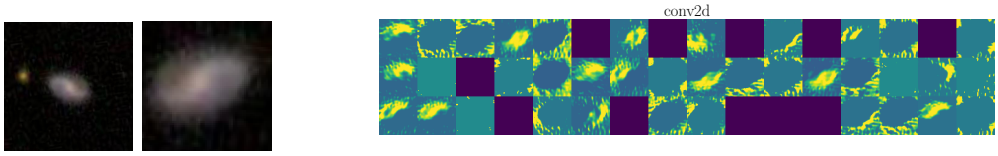
(a) Model prediction: Elliptical (41%). "True" label: Spiral (56%). ID: 100008



(b) Model prediction: Spiral (54%). "True" label: Spiral (83%). ID: 100765



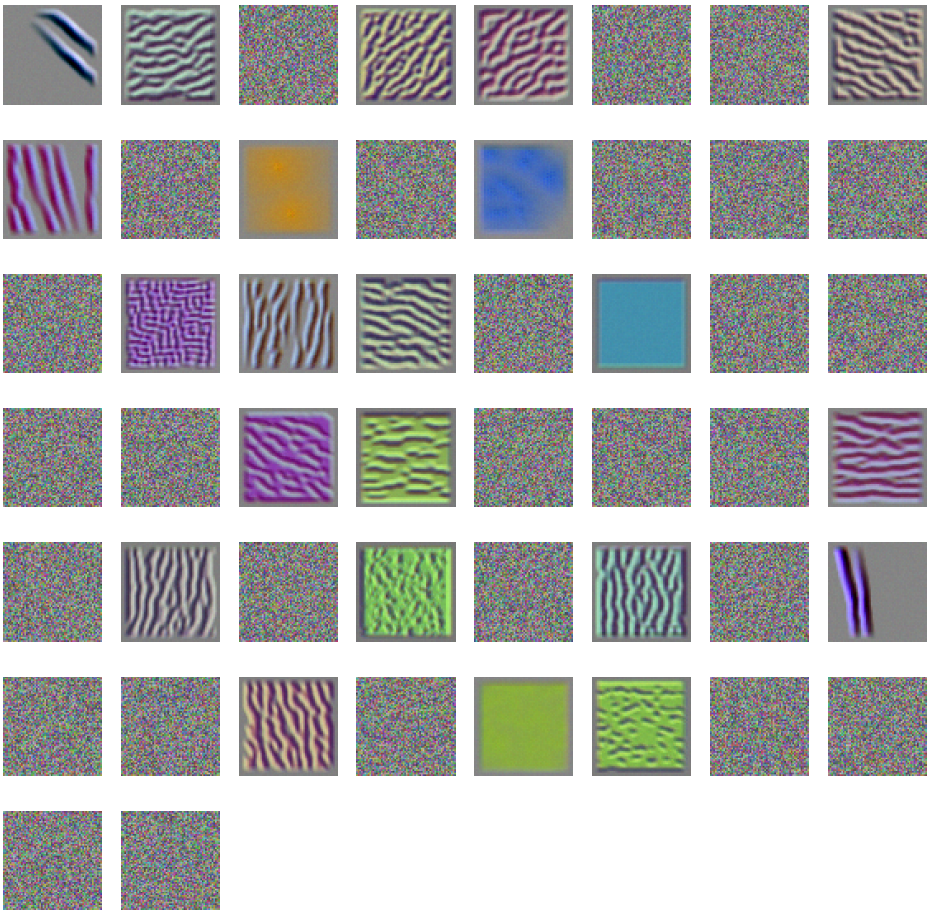
(c) Model prediction: Barred spiral (59%). "True" label: Edge-on (78%). ID: 100263



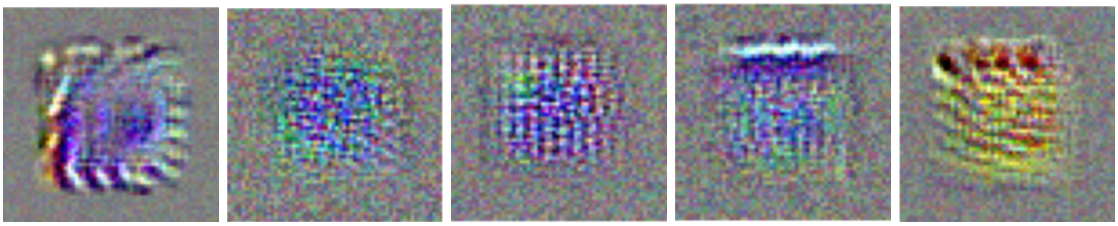
(d) Model prediction: Barred spiral (52%). "True" label: Spiral (59%). ID: 101140

(e) Model prediction: Edge-on (52%). "True" label: Edge-on (73%). ID: 100143

Figure 14: Examples of model predictions for selected out of sample galaxies and first layer CNN activations. We included examples where the model fails on purpose because there's more to learn from it. The first image is the original, full-size image from the dataset while the second is the image after cropping, downsampling and data augmentation.
We observe that the spiral in 14a is misidentified as an elliptical. The spiral in 14b is correctly identified. 14c presents an interesting case because the frame contains two galaxies, one of which is clearly edge-on. The model seems to be confused by this as it predicts a barred spiral.
In 14d, we see a spiral that's mistakenly identified as a barred spiral. 14e shows an edge-on that's correctly identified.
We see that for all galaxies, some filters seem to be "dead", i.e. identically zero. This might indicate that model complexity could be reduced without sacrificing performance.



(a) First convolutional layer



(b) Third convolutional layer (examples)

Figure 15: High mean activation input images for the filters in the first 15a and last 15b convolutional layer. Top layer images contain mostly high-level features while low level images are more detailed. Both layers contain dead filters (here: noisy images). We can identify filters in the top layer that might correspond to detecting bars, spirals, diffuse patterns (ellipticals) and dots. The last layer shows similar behaviour. The edges of the images in the last layer do not converge away from noise, possibly indicating that the network essentially ignores image edges.

4 Conclusion

In this report, we presented our steps taken in tackling two very different machine learning tasks of very different complexity. For the lower complexity problem of predicting superconductors' critical temperatures, we achieved close to state of the art results on the same dataset despite using a simpler approach than those taken by authors of, for example [8] or [16]. We identified which atomic quantities appear to correlate most strongly with a high T_c.

For the galaxy classification problem, we achieved less impressive results despite simplifying the task by a significant amount. We explained why this may be the case, attributing it mostly to inherently diffuse labels, and inspected the filters learned by the network, speculating on what features these filters were trained to identify.

References

[1] G. Czycholl. *Theoretische Festkörperphysik*. Springer Berlin, Heidelberg, 2008. DOI: 10.1007/978-3-540-74790-1.

[2] Kam Hamidieh. *Superconductivty Data*. 2018. DOI: 10.24432/C53P47. (Visited on 02/14/2024).

[3] National Institute for Materials Science Materials Database Group. *MDR SuperCon Datasheet*. 2022. DOI: 10.48505/nims.3739. (Visited on 02/14/2024).

[4] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830. DOI: 10.48550/arXiv.1201.0490.

[5] Tom O'Malley et al. *KerasTuner*. https://github.com/keras-team/keras-tuner. 2019.

[6] François Chollet et al. *Keras*. https://keras.io. 2015.

[7] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[8] Valentin Stanev et al. "Machine learning modeling of superconducting critical temperature". In: *npj Computational Materials* 4.1 (June 2018), p. 29. DOI: 10.1038/s41524-018-0085-8.

[9] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[10] Edwin. Hubble. "No. 324. Extra-galactic nebulae." In: *Contributions from the Mount Wilson Observatory / Carnegie Institution of Washington* 324 (Jan. 1926), pp. 1–49.

[11] Wikipedia. *Timeline of knowledge about galaxies, clusters of galaxies, and large-scale structure*. URL: https://w.wiki/9KSQ (visited on 02/28/2024).

[12] Kyle W. Willett et al. "Galaxy Zoo 2: detailed morphological classifications for 304122 galaxies from the Sloan Digital Sky Survey". In: *Monthly Notices of the Royal Astronomical Society* 435.4 (Sept. 2013), pp. 2835–2860. DOI: 10.1093/mnras/stt1458.

[13] Kaggle. *Galaxy Zoo - The Galaxy Challenge*. Dec. 21, 2013. URL: https://www.kaggle.com/competitions/galaxy-zoo-the-galaxy-challenge/ (visited on 02/28/2024).

[14] Paul Gavrikov. *visualkeras*. https://github.com/paulgavrikov/visualkeras. 2020.

[15] Francois Chollet. *Visualizing what convnets learn*. May 29, 2020. URL: https://keras.io/examples/vision/visualizing_what_convnets_learn/ (visited on 02/28/2024).

[16] B. Roter and S.V. Dordevic. "Predicting new superconductors and their critical temperatures using machine learning". In: *Physica C: Superconductivity and its Applications* 575 (2020), p. 1353689. DOI: https://doi.org/10.1016/j.physc.2020.1353689.