Chapter 1

Healthy-Faulty Signal Classification

In the second part of this thesis, we will apply the complex-valued deep learning framework, developed in part 1, to a real world problem of condition monitoring in industrial applications.
Condition Monitoring is defined as *the process of monitoring a parameter of condition in machinery (vibration, temperature, etc.), in order to identify a significant change which is indicative of a developing fault.*
In our technological and industrial society, the use and mastery of condition monitoring turned out to be of extreme importance, since it allows to be schedule maintenance, or other actions to be taken in order to prevent consequential damage and avoid its, often expensive, consequences. Furthermore, it can helps in increasing the lifespan of many engines, preventing faults that could develop in major failures.
Condition Monitoring techniques are normally used on rotating equipment, auxiliary systems and other dynamic machinery (compressors, pumps, electric motors, etc.) while for static devices, usually, periodic inspections are sufficient.
In our analysis, we will propose a modification (or, better, an extension) to the existing approach to the strategy that is usually followed for rotating machines, i.e. **vibration analysis**.

What engineers do, is usually taking measurements on machine bearing casings with *accelerometers* to measure the casing vibrations, sometimes provided also of electric transducers able to directly observe the rotating shafts and detect their radial (and axial) displacements. Data collected are then set of vibrational signals that can be analyzed and studied to detect clues of something bad that is happening. Vibration levels can then be compared with some historical baseline values (a "ground truth") derived after long periods of experiments, and in some cases compared with established standards such as load changes, to keep high the attention. Vibration limits can also be defined based on the machine design or components, knowing their fault frequencies of bearings.
Interpreting the vibration signal obtained is an elaborate procedure that requires specialized training and experience. With the technology progresses and increasingly complicated machines, simple comparisons and vibration limits are not anymore sufficient to guarantee the performances and accuracies needed by the companies. Luckily, also the techniques available have widely improved: thanks especially to the recent advances of machine and deep learning, signal analysis and classification is strongly simplified. Several state-of-the-art approaches have been developed, able to provide the vast majority of data analysis automatically, retrieving information instead of raw data.
In general one commonly employed technique is to examine the individual frequencies present in the signal. These frequencies, in fact, can correspond to certain mechanical components or specific malfunctions (e.g. unbalance or misalignment). So, by analyzing these frequencies and their harmonics, a specialist can in principle be able to identify the location or the type of a problem, sometimes even the cause. And this is possible weeks, even months, before the effective failure or damage of the apparatus, giving ample time to the technicians for replacing or repairing the machine.
But the interesting part in this approach is that we have a mathematical instrument that allow us

to move from the time to the frequency domain, with all the consequent advantages: the **Fourier Transform**. We will see, in the next section, that this operator not only permits the analysis of the frequency spectrum of the signal, but also represents the connection among the real-valued measurements of physical quantities and the complex-valued domain in which we can effectively put to test the framework we have developed.
It is however better to specify that frequency analysis tends to be most useful on machines that employ rolling element bearings and whose main failure modes tend to be the degradation of those bearings, which typically exhibit an increase in characteristic frequencies associated with its geometries and constructions.
Tons of different "manual" analysis are possible for those kind of signals in the frequency domain, a study of its phase spectrum for example, but in this thesis we are not really interested in them since we will rely on modern deep learning approaches, providing also efficient alternatives to the actual state-of-the-art techniques.

1.1 State-of-the-Art

In chapter **??**, we were complaining about the fact that literature does not provide almost any kind of complex-valued dataset to effectively test the models we have developed. There, we were forced to "manually construct" our data, with simple distributions of points satisfying certain properties **??**, or clever modifications of existing real collections **??**. Relying on them could have been fine for that preliminary analysis, mainly focused on the convergence and the stability of the training process, rather then on the final performances. However, we believe that, in order to prove the efficacy of our method, we should stress our complex-valued architectures on more concrete and reliable datasets.
But, is there the possibility of measuring in some way physical quantities that are inherently complex-valued? From this point of view, the answer is yes, and we just need to think about magnetic momenta. The PHD thesis we have based our work on **??**, is constructed exactly on these kind of data: Magnetic Resonance Images contains, in fact, much information in their phase, that, with a complex neural network the author managed to outperform many existing approaches that used to discard such information.
But we should not limit our horizons when we have a powerful instrument like the Fourier transform $\mathcal{F}$, that allow us to associate a complex-valued frequency domain representation to a function of time. In this new representation, the magnitude represents the amount of a certain frequency present in the original signal, while the argument is its phase offset with respect to the basic sinusoid. All of this means that, given any dataset composed of waves in the time domain, the same can be studied in the frequency domain exploiting the new power of complex-valued models, keeping all the physical information without need of expedients like before. Several papers have been published, suggesting complex-valued models to study electric, seismic **??** and vibrational signals in the complex domain. So, in principle, for any signal registered in the time domain, we can recover, and maybe exploit, the phase information thanks exactly to the Fourier transform.
Studying the problem in the frequency domain is a quite common approach to signal processing, since many operations in the time domain have a complex counterpart that sometimes turns out to be easier to perform (eg. differentiation and convolution). Furthermore, also from a purely theoretical point of view, several concepts are easier to derive and understand with complex notations.

1.1.1 Baseline Machine Learning

But how can we effectively distinguish among vibration signals? According to the traditional machine learning approach we should rely on some hand-crafted features extractable from the wave in the time domain; those features can then face a dimensionality reduction step (PCA, t-SNE, etc.) and then sent through baseline classifiers like Support Vector Machines or Multi-Layer Perceptrons. This was the fundamental strategy for years: in term of memory and time complexity it is very efficient, because each sample is reduced to a handful of meaningful values, but, in general, it lacks of generalization capabilities, since the high-level features to be used are chosen every time by an expert, depending

on the situation, and also from the point of view of the accuracy reached, more modern technologies employing deep learning are already able to outperform it.

1.1.2 Study in the frequency domain

However, even if condition monitoring is not one of the main tasks over which machine learning researchers focused mostly in the last years, several techniques and improvements could be recovered and re-adapted from the popular area of audio applications. In the end, in fact, sounds are vibrations of the air, and so their analysis can present interesting similarities with our problem.
It is exactly from the widely developed theory of deep learning applied to audio processing, that we inherit techniques suitable to study a wave signal in the frequency domain. The computational instrument that allow us to do so is the **Short-Time Fourier Transform** (**STFT**). Discrete STFT is a method of partitioning continuous signals (in the time-dependent representation) over a long period into shorter segments (usually overlapped) at short time intervals, and applying a Fourier transform to each of those segments. An additional step can be taken, at this point, deriving the power spectrum of the signal: taking the square modulus of the STFT we can construct a 2D sample, i.e. an image, representing the amplitude of the signal for a particular frequency at a particular time.
In figure 1.1 we took a random signal from the dataset we are going to study, and we transformed it in the Fourier representation: the leftmost picture represents the original wave, the central one is the result after the application of the Fast-Fourier transform (so the signal in the frequency domain), while the rightmost is its power spectrum. After the computation of the power spectrum for each

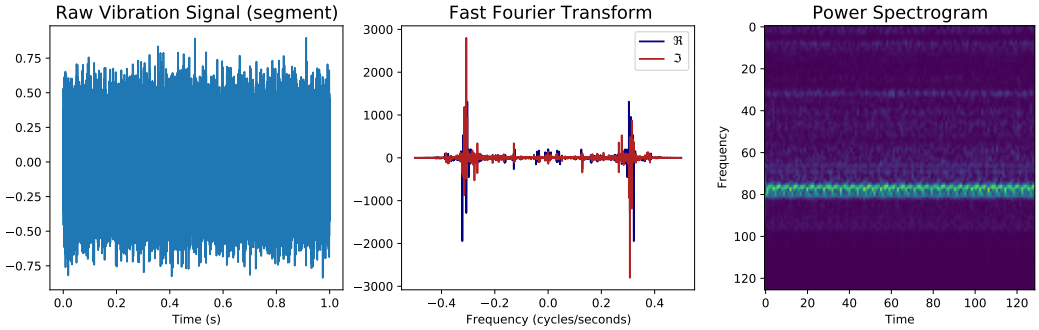

Figure 1.1: Representation of a vibration signal in the time domain (left), in the frequency domain via FFT (center) and with its power spectrum (right).

vibration signal at this point, there are two directions that the researcher can decide to follow:

- keeping the two-dimensional input and building a machine learning classifier able to work directly with the spectrograms (usually a Convolutional Neural Network);
- taking a further step and extracting from the spectrogram the so called *Mel Frequency Cepstral Coefficients* (MFCC): these coefficients, widely used in speech recognition tasks, permit an efficient compression of the "important information" carried by the signal, and can be simply given in input to an ordinary classifier.

Even if they were not properly designed for vibration analysis applied to condition monitoring, the usage of MFCCs can bring very good results together with an high efficiency in term of space and time complexity, because of the information compressed.
In figure 1.2 are represented schematically the two main state-of-the-art approaches (high-level features extraction vs power spectrograms) for healthy-faulty vibration signal classification, that we just described.
In the successive analysis, we will propose an alternative to the approaches listed above, that is more suitable to complex-valued deep learning: the coefficients of a Fourier transform are, in fact, inherently complex, and so, instead of losing all the phase information taking the power spectrum (that is constituted only by their amplitudes), we stop one step before keeping the original complex-valued variables. In the end, deriving the complex spectrum is equivalent to computing a discrete Fourier transform over short overlapping windows of the signal.

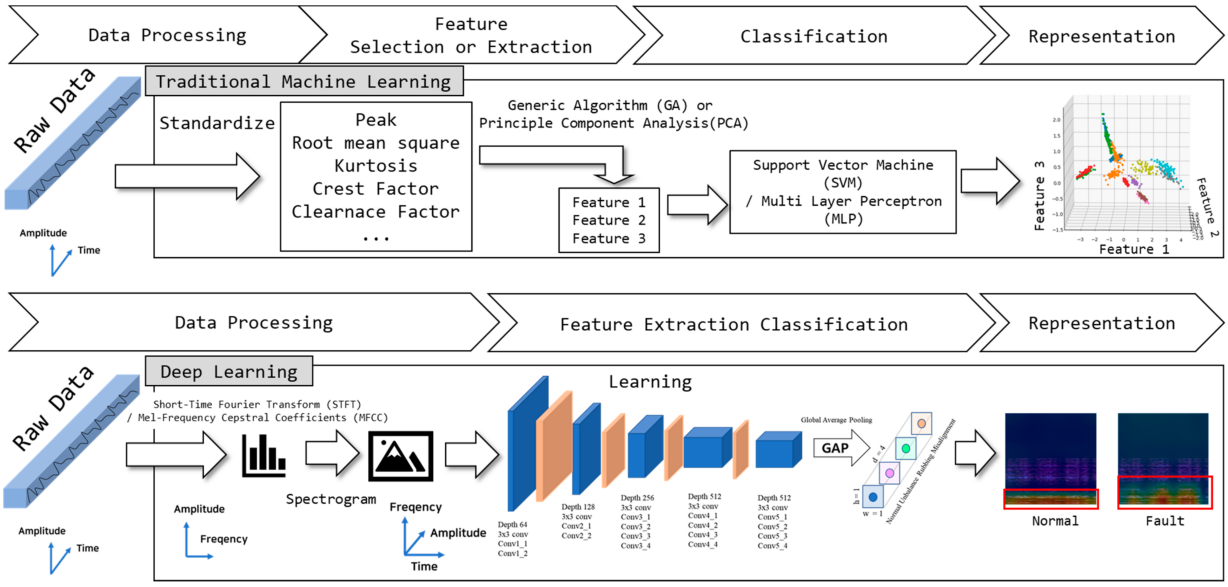

Figure 1.2: State-of-the-art approaches for vibration data classification applied to predictive maintenance (source: [**?**]).

1.2 Bonfiglioli

The main motivation of this thesis work was the development and testing of an unconventional deep learning approach, based on complex-valued neural networks, on a series of datasets provided to FBK by Bonfiglioli 1.3, a worldwide designer, manufacturer and distributor of a complete range of gearmotors, drive systems, planetary gearboxes and inverters.

The company provided a large number of datasets collected from as many experiments, with the purpose of simulating artificially the most frequent faults, that usually occur over the bearings of different kind of engines. Our objective will be then to find an efficient approach for failure detection and classification.

Figure 1.3: Bonfiglioli logo.

1.2.1 Simulation Environment

Data have been collected at the Bonfiglioli headquarters of Rovereto (TN), with the experimental setup schematized in figure 1.4. The laboratory is constituted of four different workbenches in which different experiments can be conduced. In each workbench, a lab-scale rotating simulation equipment is installed, constituted by two working axis, one representing the healthy configuration while the other will be modified in order to simulate a damaged engine, in one of its components. Tons of sensors have been collocated all around the mechanical system in order to measure physical quantities of interest (vibration, temperature, current, etc.) for both the *healthy-axis* and the *faulty-axis*.
So far, in the experiments, Bonfiglioli managed to reproduce up to five different kind of faults, that were artificially simulated using mechanical tricks:

- *unbalance*: the rotor of one of the engines was unbalanced via two masses of 3g;
- *bearing damage*: not specified in their reports, as they just states that bearings have been damaged radially;
- *solar*: the solar of the gear reducer was damaged via an electric pen;
- *pinion*: the input bearing of the gear reduced was damaged;
- *crown*: the crown of the gear reducer was damaged via an electric pen.

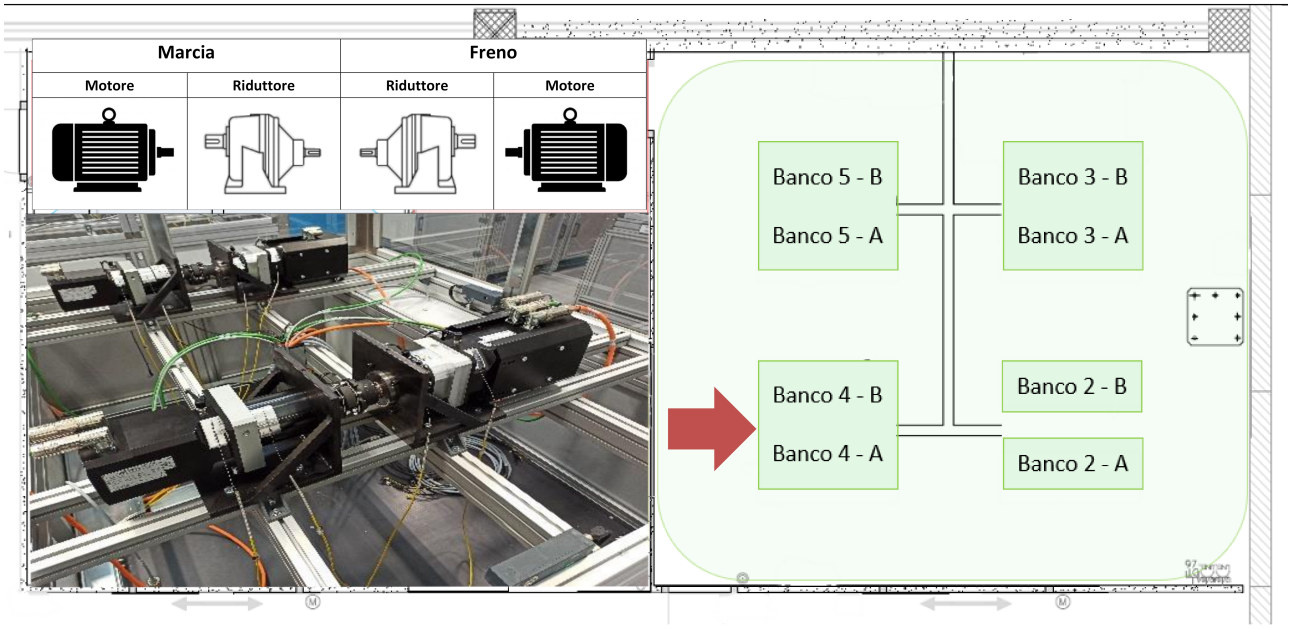

Figure 1.4: Schematic representation of the experimental workbench used by Bonfiglioli.

Just to provide a visual point of view, in figure 1.5 we reported the images of some simulated faults. In order to have data more heterogeneous, able to represent the engine behavior under different con-

Figure 1.5: Some of the faults simulated by Bonfiglioli on the mechanical components of the system.

ditions, and to improve the generalization capabilities of future models trained over them, Bonfiglioli setup a *working cycle*, during which measurements were taken. This cycle is periodic with a period of 24 hours, in general repeated up to five times to collect most of the datasets. It is characterized by the engine varying its speed and torque with a quite regular behavior following a piecewise constant function, also to guarantee stationary conditions during the acquisition phases. Data were not, in fact, collected continuously during this cycle: 30 or 60 seconds long windows were fixed along this period, during which sensors were actively measuring. An example of working cycle has been reported in figure 1.6. Vibrations signals, the ones we are mainly interested in, were collected thanks to accelerometers attached to the components of the engine (motor, brake and multiplier) they wanted to test; as explained above, each acquisition was 30/60 seconds long, with a sample rate of 25600 Hz: sampling was performed because a vibration is a periodic signal in the time domain, and most fault signals have periodicity too. Therefore, sampling can be used to examine the consistency and continuity of each condition [**?**]. The signal segmentation for sampling is usually based on the rotational frequency of the rotor. Generally, in fact, in rotating equipment, the rotational frequency is the most dominant component, and the majority of fault contributions appear in its harmonic form.

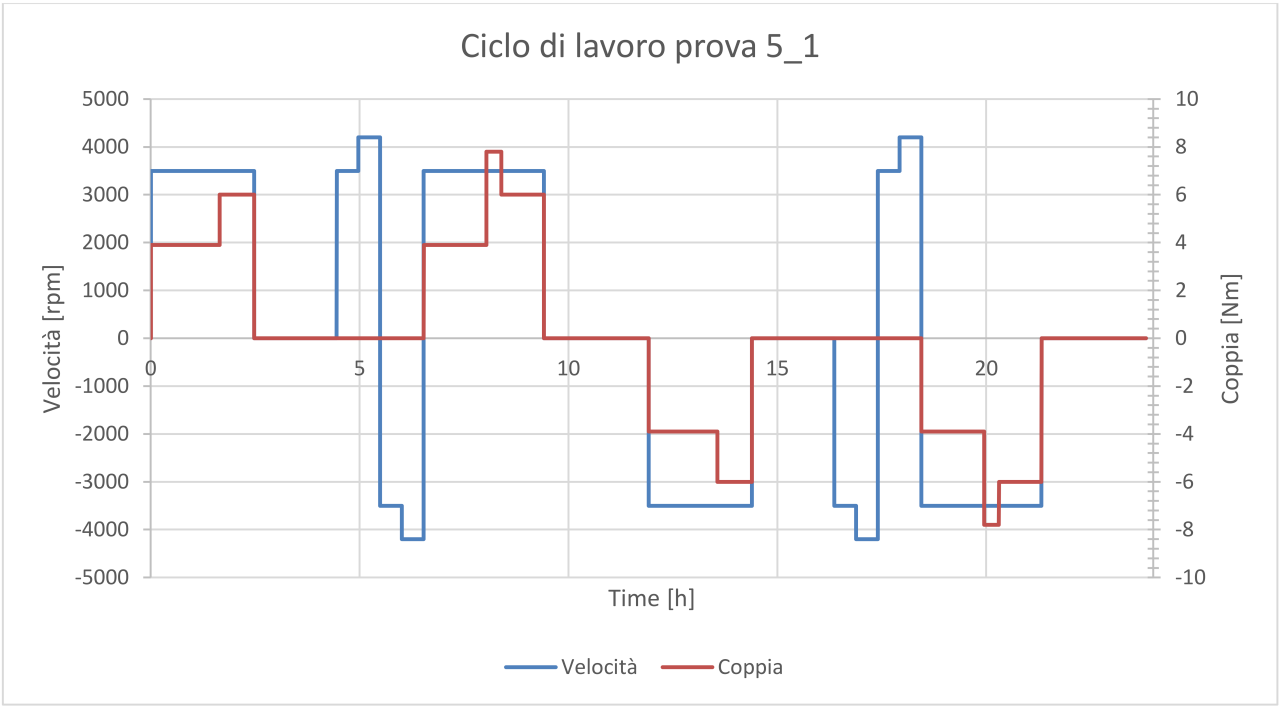

Figure 1.6: Example of working cycle (from Prova-5) of the Bonfiglioli experimental setup.

1.2.2 Datasets

In these last months, the Bonfiglioli laboratory have setup different experiments, in order to provide data relative to different kind of engines and for different types of simulated faults. For each of those runs, containing 120 hours of measurements, a dataset in `.h5` format has been provided, with the values registered during a working cycle. Those datasets had an internal structure quite mazy, and so we had to rely on a custom library, provided by FBK researchers, to load and process the data. Each sample contains several entries, related to the different variables measured during the process: beyond vibration signals collected through the accelerometers, Bonfiglioli decided to monitor also the temperature of some components, voltage and current flowing through them, together with the phase information of the inverter. Additionally, for the experiments realized more recently, they used an additional component in their setup, a *multiplier*, providing a second vibration signal.
For what concern our work, we are interested only in the vibration signals. In table 1.1 are reported, with some technical details, the dataset we have used in our analysis.

N	Name	Subname	Engine A	Duration (h)	Fault
4	Prova 4 - BN90 - HF Sbilanciato	4_2	BN90	120	Unbalance
5	Prova 5 - MP105 - HF Cuscinetto	5_3	BMD118	120	Bearing
6	Prova 6 - BN90 - HF Cuscinetto	6_3	BN90	100	Bearing
10	Prova 10 - MP105 - HF Solare	10_1	BMD118	120	Solar
10	Prova 10 - MP105 - FH Solare	10_4	BMD118	120	Solar
11	Prova 11- S30 - HF Cuscinetto	11_1	BN112	120	Bearing
13	Prova 13 - S30 - HF Pignone	13_1	BN112	120	Pinion
15	Prova 15 - S30 - HF Corona	15_1	BN112	78	Crown
15	Prova 15 - S30 - HF Corona	15_2	BN112	48	Crown

Table 1.1: Summary of the main technical features of the datasets provided by Bonfiglioli and used in our analysis.

1.2.3 Baseline Classification Approach

For completeness, we would like to briefly describe also the baseline approach, the one actually followed by FBK researchers, even if in this work it wasn't either considered.
The procedure actually implemented in the project is quite similar to the one presented as the state-of-the-art. The idea behind is, in fact, fundamentally the same: given a dataset of vibration acquisitions, these are splitted into subsignals of a given fixed size (usually 3 seconds), preprocessed and sent through a "feature extractor" that takes from them the information useful for the classification. Different kind of extractions are possible: up to now you can choose among MFCCs, ordinary Short Time Fourier Transform and power spectrograms, all of them with a wide margin of customization. The main library used for this is feature-extraction part is `scipy`.
All the information got thanks to this step is then sent through a neural network classifier, that in this case is a Residual Network [1], with a fundamental block constituted by two $3x3$ convolutions and a few regularization layers.
With respect to the procedure we are going to implement, this is much more efficient from the point of view of the memory complexity (especially if you select the MFCCs), but it requires an architecture that is much larger. We will prove that we can still manage to obtain very good results even with a much smaller number of parameters.

1.2.4 Complex Spectrogram Classification

As anticipated, the approach we decided to follow is slightly different. In sound classification tasks, the advantage provided by convolutional neural networks has been widely demonstrated, in recent years, providing a valid alternative to the standard procedure of exploiting high-level features (like MFCCs) [].
In our implementation, we first split the data acquisitions into signals of fixed length: we found that a window of 1.0 second was representing a nice tradeoff between dataset size and resolution, preserving, at the same time, the eventual periodicity of the wave. We then proceed computing, for each signal in the dataset, the corresponding **complex spectrogram** (essentially a STFT), i.e. a plot associating amplitude and phase in a coordinate system that foresees the time on the horizontal axis and the frequency in the vertical one. In the end, instead of losing all the phase information considering the power spectrum, we keep the complex data and exploit the new complex-valued deep learning framework we have just developed. Complex convolutional networks, in particular, seems to suggests that a model sensitive to the phase structure could provide very nice results.
From a practical perspective, the spectrogram extraction step was preformed using the corresponding functions provided by `Torchaudio` (and based on `librosa`): we believe that their implementation is better, at least for classification purposes, with respect to its alternative that relies on `scipy`, mainly because of efficiency and normalizations reasons.
The main difficulty in this implementation consists into finding a proper resolution, both in time and frequency, for the signal to effectively point out its discriminative characteristics: we found that, for a 1 second long vibration, setting the `n_fft` parameter to 300 and leaving the other to their default values was guaranteeing nice results. As a general rule, however, we feel to empirically suggest that a nice combination of parameters is the one creating a spectrogram that is square, or at most rectangular with longer time dimension, since, apparently, these cases were the ones behaving better with our models.
In figure **??** we left an example of the data we are going to classify. Since complex data are quite difficult to plot, we relied on a modified "colorize" function, that inherits from `mpmath` a mapping from the polar representation (m, θ) of complex numbers to triplets of colors in the `hls` representation. We know that the result is not very good aesthetically, but that's what we could achieve in continuity with amplitude and phase spectra.

[1] ResNet

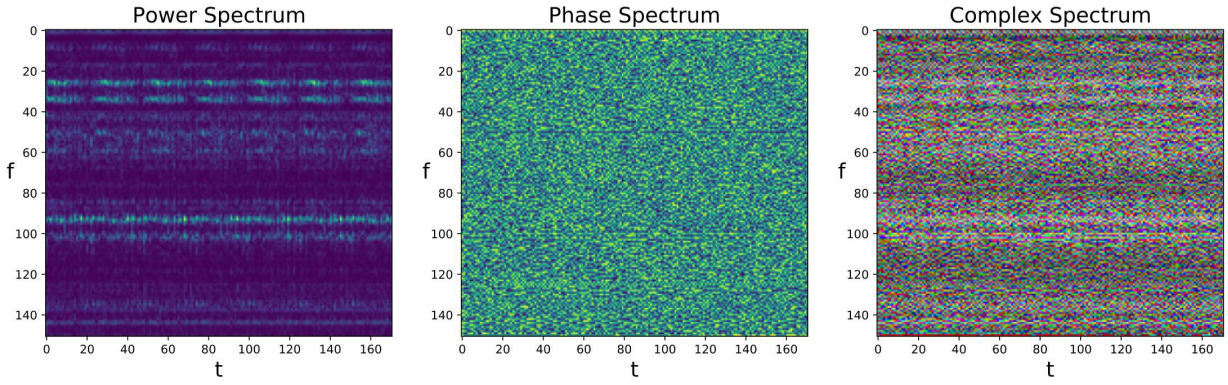

Figure 1.7: Example of spectrograms derived from a signal in the Bonfiglioli dataset. For completeness, we represented also the power spectrum (right) and the phase spectrum (center). The leftmost one is instead our representation of the complex spectrum.

1.3 Bonfiglioli - Results

Before presenting the effective results of this classification task, a couple of words should be spent presenting the analysis setup. As for the PhaseMNIST dataset **??**, we decided to employ two different networks: a purely complex-valued architecture and its corresponding real-valued counterpart, that treats real and imaginary parts of the input as two independent channels and, for this reason, has twice the trainable parameters of the first. Those architectures are based on the canonical forms of Alexnet and VGG16, with a few convolutional layers stacked on top of a multi-layer perceptron. In figure **??** we can visualize the models we have effectively used. It is interesting to notice that the networks are quite small (1092 and 2188 parameters, respectively), and the reason is that ...

Also, an important role is covered by the 3×3 average pooling layers, that are fundamental to reduce the dimensionality of the inputs. However, the only important difference among the complex and the real architectures, beyond the data type employed, is the activation function: `cardioid` for the first and ordinary `ReLU` for the second one.
Regarding the training setup, we followed a quite ordinary approach: since this is a classification problem we relied on the **crossentropy** (on the output's magnitude) as loss function and on the **categorical accuracy** to evaluate the performances of a model. The optimizer is a complex-valued version of `Adam` with an exponentially decaying learning rate (to improve the convergence at the beginning and the precision in the late) following the rule $lr(x) = 10^{-2} * (0.9)^{(x/100)}$. No further data preprocessing was needed (especially because spectrograms are already normalized) and labels were one-hot encoded. We used a $75\% - 25\%$ splitting to determine training and test sets.
The first results the we want to present are the classification scores obtained over the datasets listed in table 1.1. For all of them we extracted the vibration signals measured by the accelerometers installed on the engines, that are divided into two classes, "healthy" (label 0) and "faulty" (label 1), depending on the axis they belonged to. Recall, also, that for some of them (basically all but 4_2 and 6_3) we could use two signals (motor and multiplier) rather then just one (motor only): in order to verify that the classification could effectively benefit of multiplier vibrations, for those datasets we repeated the analysis twice, one with and one without this kind of measurements. We then proceed extracting the complex spectrograms from those signals: setting the segmentation window to 1.0 second and the parameter `n_fft` to 300, we obtain a dataset of images of dimension $(1, 151, 171)$ (just like the ones in figure 1.7). In the cases with the multiplier, we simply consider the two spectrograms as two independent channels of the same image: in this way, the inputs will have shape $(2, 151, 171)$.
In table 1.2 we summarized the accuracy and loss obtained for all the single validation datasets provided by Bonfiglioli, via binary classification for 100 epochs.

Dataset	Samples	Fault type	Accuracy (%)	Loss ($\cdot 10^{-3}$)
4_2	3508	Unbalance	100 - 100	0.014 - 0.052
5_3	6071	Bearing	99.61 - 99.61	7.311 - 8.481
5_3 w/ multi	6071	Bearing	99.80 - 99.74	5.014 - 7.398
6_3_1	700	Bearing	100 - 100	0.018 - 0.042
6_3_2	2102	Bearing	100 - 100	0.004 - 0.008
10_1	5966	Solar	99.40 - 99.53	6.895 - 6.300
10_1 w/ multi	5966	Solar	99.48 - 99.41	6.710 - 7.638
10_4	5979	Solar	100 - 99.22	3.445 - 10.345
10_4 w/ multi	5979	Solar	100 - 99.04	5.142 - 11.538
11_1	3744	Bearing	100 - 100	0.001 - 0.002
11_1 w/ multi	3744	Bearing	100 - 100	0.001 - 0.028
13_1	3750	Pinion	100 - 100	0.008 - 0.014
13_1 w/ multi	3750	Pinion	100 - 100	0.001 - 0.007
15_1	2511	Crown	100 - 100	0.001 - 0.040
15_1 w/ multi	2511	Crown	100 - 100	0.039 - 2.025
15_2	1498	Crown	100 - 100	0.027 - 0.937
15_2 w/ multi	1498	Crown	100 - 100	0.056 - 0.164

Table 1.2: Binary classification of single Bonfiglioli datasets (real - complex).

1.3.1 Multiclass Classification

After performing all the binary classification tasks on the single datasets provided, is time to verify if our model are effectively able to distinguish also among different types of fault.
We setup, then, a multiclass classification problem, with 6 possible labels: *healthy*, *unbalance*, *bearing*, *solar*, *pinion*, *crown*. In order to have a properly balanced dataset, with all those classes being equally represented, and also to reduce the memory complexity of this task, we decided to adopt only one cycle of measurements for each dataset considered, i.e. instead of taking all the 120h of 4_2 we took only the first 24h. And this is a reasonable solution to guarantee equilibrium in the distribution of labels, given the heterogeneous collections of data provided. The equity, in reality, is not total: each dataset is composed by one half of healthy signals, and so this class will be represented, in the final distribution, five times more respect to the others. We could have bypassed this problem simply by taking subsets of healthy data every time, but, apparently, the results have not been affected so much, and so we left the proportions as they were. In the end this "comprehensive" dataset is constituted by 4612 samples (complex spectrogram).
Back to the classification, the training setup and the architectures are essentially the same as before: the only difference is that now we have 6 different classes, and so the last layer of both networks should then have 6 units rather than 2. In figure 1.8 we can observe and analyze the training loops'

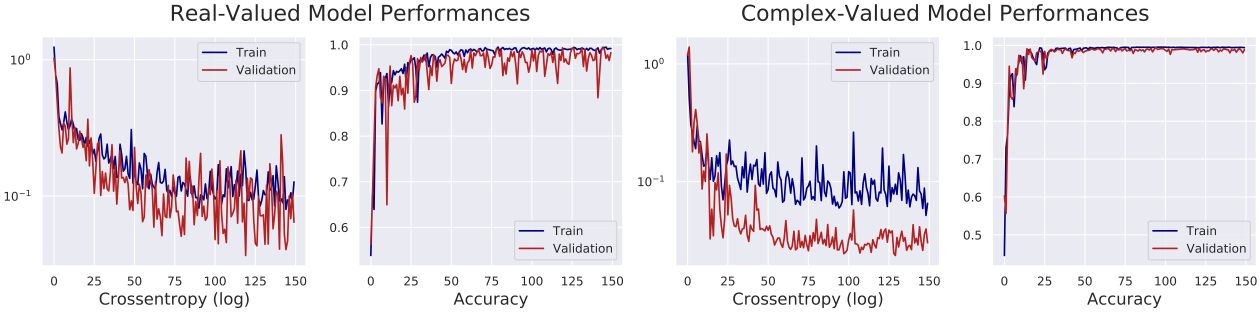

Figure 1.8: Training loops of real and complex-valued models for the Bonfiglioli multiclass classification problem.

scores of both the real and the complex-valued models over this comprehensive dataset. Differently from the situations before, this time it seems that something emerged. The first contrast that comes

to our eyes is of course the stability: for the complex network we see the train and validation losses properly converging together, while, the real case, especially for the test line, is characterized by a lot of fluctuations. In the end, however, we see that the final performances are not so different: in table **??** we reported a few statistical estimators of the results, while in figure 1.9 we can check the confusion matrices of both models. Just like before, we are nearby the perfect classification, with all

	Accuracy	**Precision**	**Recall**	**F1Score**
Healthy	0.986 - 0.990	1.000 - 0.988	0.973 - 0.992	0.987 - 0.990
Unbalance	0.991 - 0.999	0.900 - 0.989	1.000 - 1.000	0.947 - 0.994
Solar	0.997 - 0.994	1.000 - 0.954	0.979 - 1.000	0.990 - 0.977
Bearing	0.992 - 0.997	0.941 - 1.000	1.000 - 0.972	0.969 - 0.986
Pinion	1.000 - 0.997	1.000 - 1.000	1.000 - 0.964	1.000 - 0.982
Crown	1.000 - 0.997	1.000 - 1.000	1.000 - 0.966	1.000 - 0.983

Table 1.3: Statistical estimators of the Bonfiglioli analysis (real - complex).

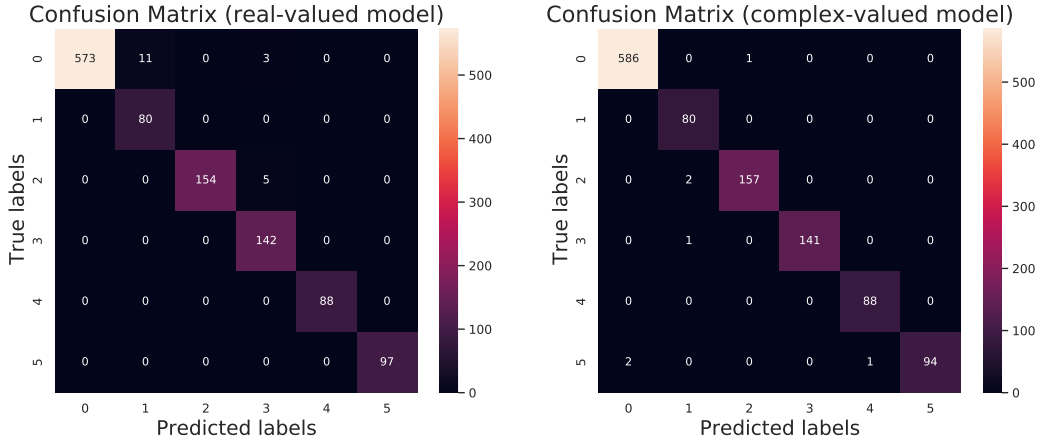

Figure 1.9: Confusion matrices of the Bonfiglioli multiclass classification problem.

the statistics almost reaching saturation. For this reason we believe that in this kind of situations, a qualitative analysis of the process' stability can be the better discriminator among two equivalent approaches.

1.3.2 Class Activation Maps

Several works have shown that the units and filters of convolutional layers in neural networks actually behaves as real object detectors, despite no supervision on the location of those objects was provided. However, this ability is lost once the input is flattened to be sent through fully-connected layers, that is what happens also in our architectures. In order to preserve the locality of features learned by the convolutions, an interesting technique have been developed. First of all, we need to remove from the network all the linear layers, leaving the last convolution in direct contact with the output layer (that has as many units as the number of classes of the problem). Then, to replace the flattening layer, we rely on **Global Average Pooling** (GAP) [**?**]: a method developed first as a structural regularizer, but that was then found able to retain its remarkable localization ability until the final layer. Zhou et. al. [**?**] proposed a procedure for generating **Class Activation Maps** (CAM), that are visual representation of this locality, since they allow to determine the discriminative regions used by a CNN to identify a particular category. To summarize it, global average pooling returns the spatial average of the feature map for each channel of the last convolutional layer; a weighted sum of this values is then used to generate the final output.
Formally, let $f_k(x, y)$ be the activation of unit k in the last convolutional layer at spatial location

(x, y). The GAP result for unit k is $F_k = \sum_{x,y} f_k(x, y)$. Thus, for a given class c, the input to the final layer is $\sum_k w_k^c F_k$, where w_k^c is the weight corresponding to class c for unit k. Essentially, w_k^c indicates the importance of F_k for class c. The class activation map for the class c is then defined as

$$M_c(x, y) = \sum_k w_k^c f_k(x, y)$$

Intuitively, we expect each unit to be activated by some visual pattern within its receptive field. The class activation map is simply a weighted linear sum of the presence of these visual patterns at different spatial locations. Upsampling those activation maps, we can identify, for each input sample, the local region that the network effectively use to determine the category.
Another important aspect of this approach is its extreme efficiency: removing all the fully-connected layers we are drastically reducing the number of learnable parameters, speeding up the train and increasing the flexibility of the model.
Back to our original problem, we implemented the CAM detection phase in the previous complex-valued architecture: we simply had to remove the final part (the multi-layer perceptron), and add a global average pooling. Running the training loop again with this "mutilated" architecture, we still managed to obtain a nice convergence, and very good accuracy values ($> 95\%$ for both the training and the test sets), even if the new model has just 456 parameters.
In figure 1.10 we represented the CAMs for each class of the Bonfiglioli dataset, in contrast with the average power spectrum that is sent through the model. Those images have been realized extracting, for each sample in the validation set, its class activation map, i.e. a linear combination of the final 8 feature maps multiplied by the weights vector corresponding to the predicted class.

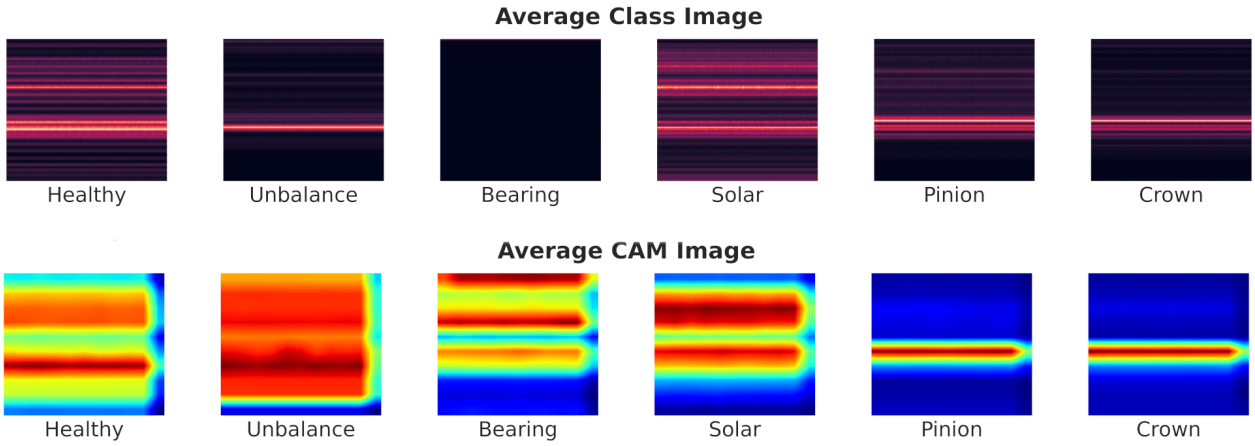

Figure 1.10: Average spectrograms and class activation maps of the Bonfiglioli validation set.

Looking at those images it is quite clear why we obtained so high performances up to now. Both the spectrograms and the activation maps are quite different from class to class, and it seems that we can easily distinguish among them, even without the help of the networks.

1.4 Mendelay Data

The last dataset we are going to analyze for this chapter is the Mendeley Dataset [**?**]: a collection of vibration signals collected from bearings with different health conditions and under different time-varying speed conditions. The problem is again an healthy-faulty classification task with two different "bad" classes: faulty with an inner race defect and faulty with an outer race defect.
The experiments have been realized over a machinery fault simulator with the setup shown in 1.11. The shafts is driven by a motor which rotational speed is controlled by an AC Drive. Two ball bearings are installed to support the shaft: the left one is healthy, while the one on the right is the experimental bearing, replaced during each experiment with a component of a different health condition. Again,

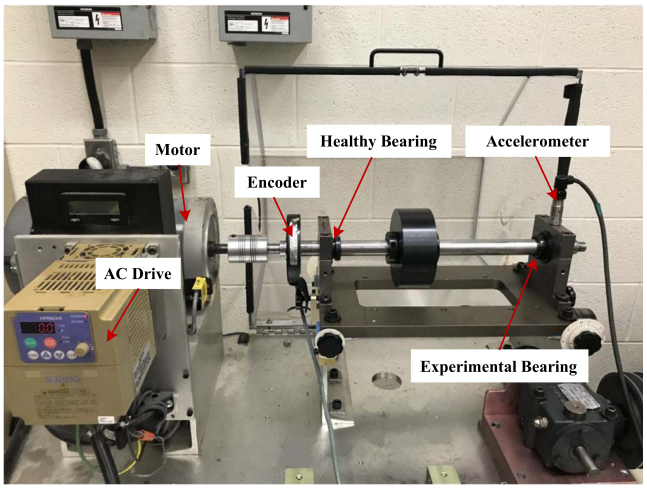

Figure 1.11: Experimental setup for Mendeley data.

vibration data are collected thanks to an accelerometer fixed on the housing of the experimental bearings, while an incremental encoder is used to collect the rotational speed values.
These datasets were collected and reorganized with the purpose of evaluating the effectiveness of methods developed for bearing fault diagnosis under time varying speed conditions. This last point is the main difference with most of vibration datasets available from actual literature, that have been collected under constant speed condition (as Bonfiglioli's). The complex-valued models we have developed so far, however, were not properly designed to work on such dynamic signals, but this still represents an opportunity to stress and push to the limit the approach we have designed.

1.4.1 Dataset Structure

Given the various general purposes for which this dataset was designed (i.e. condition monitoring and analysis of bearings' frequency characteristics under time-varying rotational speed conditions), its internal structure is slightly intricate. First of all, each sample has two channels, one for vibration data and the other for rotational speed data. Each signal is sampled at 200000 Hz and the sampling duration is 10 s. There whole collection of signals is divided into 36 different datasets, depending on two experimental settings:

- *health condition* (healthy, faulty with inner race defect, faulty with outer race defect);
- *speed varying condition* (increasing speed, decreasing speed, increasing then decreasing speed and decreasing then increasing speed).

Therefore, there are 12 different cases for configuration, each of them constituted of 3 trials, to ensure the authenticity of the data, that explain the 36 subsets. Raw signals and corresponding spectrograms are, in the end, visually similar to the ones of Bonfiglioli **??**. Data in the second channel, containing speed information, can be properly visualized via their spectrograms, looking at the frequency changes in the signal through the time (figure 1.12). For the data generation part, two parameters still needs

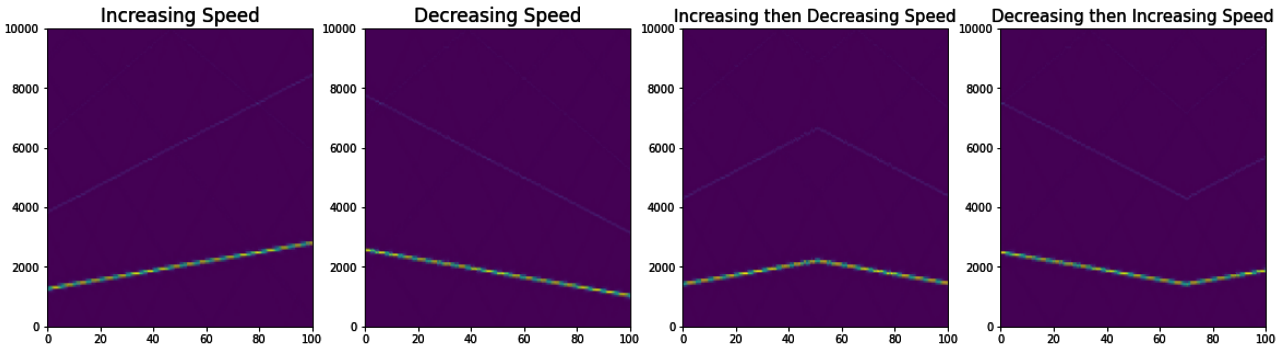

Figure 1.12: Spectrograms of the four varying-speed conditions of Mendelay data.

to be fixed:

- the `n_fft` of `torchaudio.transform.Spectrogram`, representing the resolution of the spectrogram;
- the "split rate" parameter, determining the number of subsegments in which each signal from the dataset has to be split.

The latter turns out to be quite important in this problem: recall, in fact, that each vibration registered in the dataset is 10 seconds long, with a sample rate of 200000 Hz, and so it is constituted by a total of $2 \cdot 10^6$ points, and so is quite long. Furthermore, the dataset itself is quite small: only $36 \cdot 3 = 108$ signals, number that could be increased just splitting them.

1.4.2 Mendeley Classification

We start by fixing the length of the subsignals equal to the sample rate (200000 points): as discussed in **??**, it is quite common that fault signals have periodicity, and we believe that the sampling rate provided has been chosen to properly detect this feature. Now, we have a dataset of 360, one second long, vibrations. Train-test splitting is again $75\% - 25\%$.
Even if they were not designed to deal with vibration signals registered in non-stationary conditions (with varying speed), we decided to rely on the same architecture built for Bonfiglioli analysis **??**.
We trained both models on the Mendelay dataset with all the setup described above, and the results are briefly summarized in picture 1.13. Those are very bad results: the training errors and accuracy,

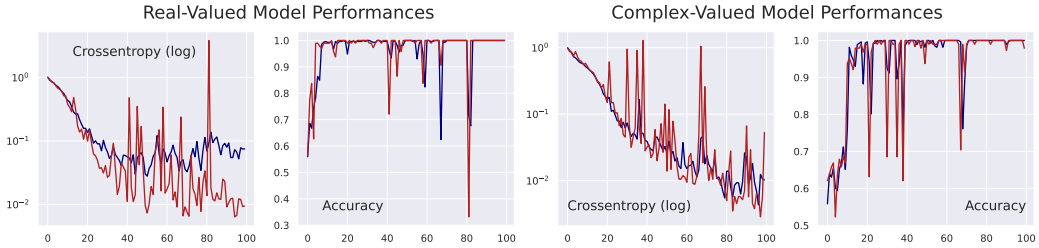

Figure 1.13: Performances achieved over the Mendelay dataset constructing one-second long sub-signals.

especially in the real-valued case, behaves nicely, but the models look like having bad generalization performances, given the high instability that we can notice on the validation lines. Now, since the architecture is already quite small, and we have already exploited the canonical techniques to prevent overfitting (normalization, dropout) what we can do is acting directly on the dataset.
Other tests realized have proved that using only the files that belong to the same speed class (increasing, decreasing, increasing then decreasing or decreasing then increasing), so keeping only monotonic or non-monotonic speed samples together, we noticed that the overfit and the instability were more limited.
Under this perspective, we believe that further reducing the length of input signals, even before constructing the spectrograms, we can manage to reduce also the effects due to high variations in rotational speed. To prove it, we repeated the train considering samples 0.25 seconds long. In this way we have much more samples (3600) in the dataset. For completeness, it is better to specify that to obtain the results shown in figure 1.14 we established a decay schedule for learning rate, in the hope of a faster, but more accurate, convergence.
Finally, this is a really nice behavior, with both models reaching the convergence and the perfect classification (100%) accuracy. This means that our supposition was probably right, and a suitable approach to signal classification in case of time-varying speed conditions, is to fragment the original wave into smaller samples to reduced the effects of this non-stationarity. Of course, we believe that a necessary hypotheses for this to happen is some kind of monotony in the rotational speed variation: also in the third and fourth cases 1.12, we could clearly distinguish two regions in which the velocity behavior was monotonic and amenable to the first two. Probably, an higher irregularity in the rotational speed would have made also the classification more unstable and challenging, but we would need further more datasets to verify it.

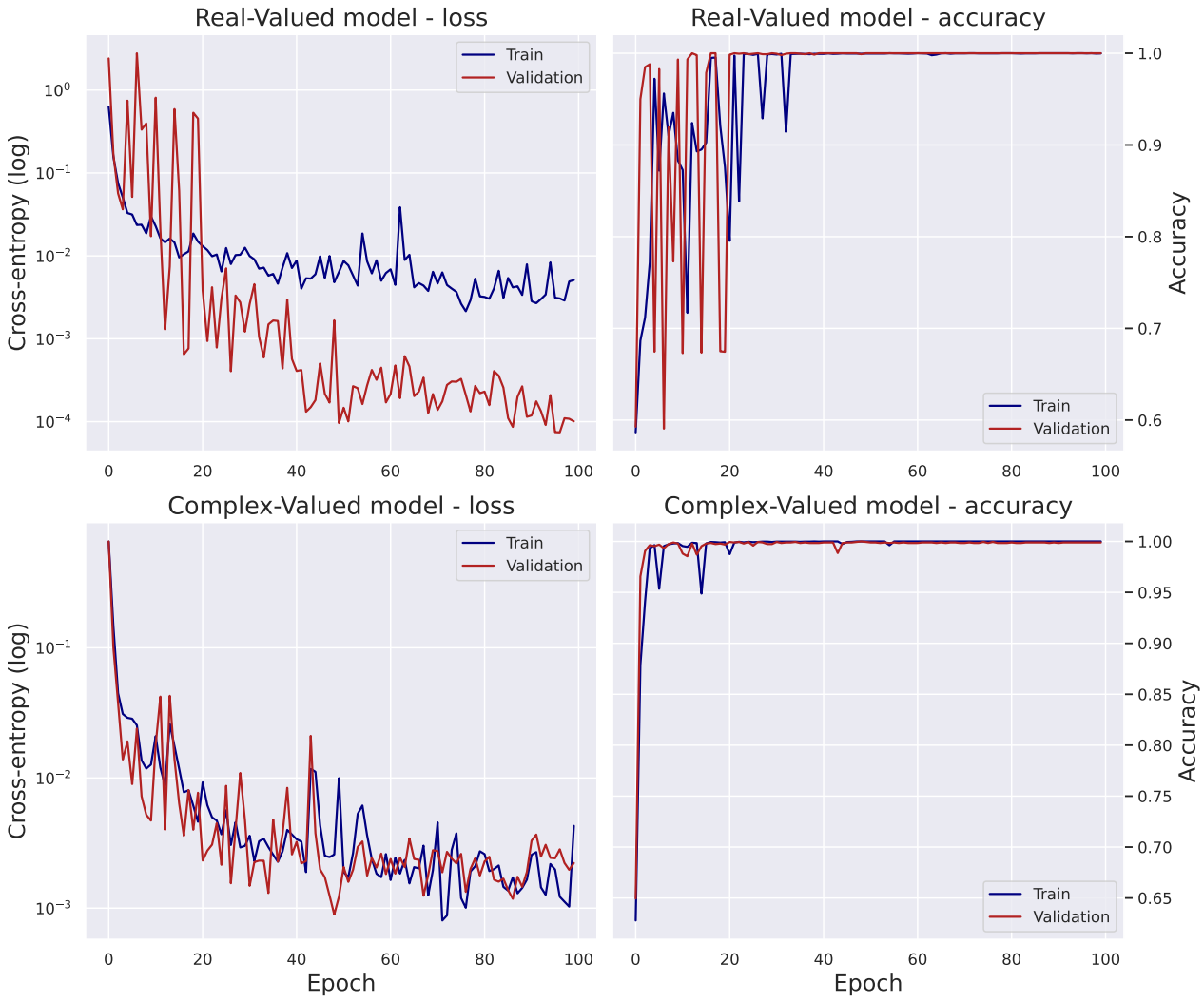

Figure 1.14: Performances achieved over the Mendelay dataset constructing 0.25 seconds long sub-signals.

To conclude this part, we want to try stressing the generalization performances of our models. This can be done by first dividing the data files based on the four speed conditions (increasing, decreasing, increasing then decreasing, decreasing then increasing), training the networks over just one of them, and computing the classification accuracy over the others. In this way we can effectively verify which is the impact of non-stationary rotational speed conditions in the training process. The results, reported

100.00	99.72	98.89	97.18	99.94	99.00	99.50	99.34
100.00	99.94	79.70	77.93	99.89	96.18	96.29	73.56
100.00	100.00	88.05	82.13	98.34	98.23	100.00	89.77
100.00	99.67	88.38	94.63	97.90	94.08	99.94	98.56

Table 1.4: Generalization performances of real and complex-valued models over the Mendelay dataset.

in table 1.4 are generally good: for almost all the combinations both networks managed to converge, obtaining nearly the perfect classification accuracy. The cells colored in red represents the cases in which the network barely overfit and wasn't able to generalize properly, with the validation loss growing instead of decreasing during the training. There are two cases in which both models failed, and one seeing the only the complex network to do so (even with a nice accuracy, in the end). What

is more interesting, instead, are the cells colored in orange: in those situations, limited to real-valued models, the loss behaved correctly, but the accuracy reached was quite limited, at least with respect to the score offered by its complex-valued counterpart. Furthermore, except for one case (the red one), all the combinations shown the complex architecture to reach higher accuracy, proving again that these kind of models are less likely to overfit and, in general, guarantees better generalization performances.