Deep Learning in the Wild — PREPRINT

Thilo Stadelmann[1], Mohammadreza Amirian[1,2], Ismail Arabaci[3], Marek Arnold[1,3], Gilbert François Duivesteijn[4], Ismail Elezi[1,5], Melanie Geiger[1,6], Stefan Lörwald[7], Benjamin Bruno Meier[3], Katharina Rombach[1], and Lukas Tuggener[1,8]

[1] ZHAW Datalab & School of Engineering, Winterthur, Switzerland
[2] Institute of Neural Information Processing, Ulm University, Germany
[3] ARGUS DATA INSIGHTS Schweiz AG, Zürich, Switzerland
[4] Deep Impact AG, Winterthur, Switzerland
[5] DAIS, Ca' Foscari University of Venice, Venezia Mestre, Italy
[6] Institut d'Informatique, Université de Neuchâtel, Switzerland
[7] PricewaterhouseCoopers AG, Zürich, Switzerland
[8] IDSIA Dalle Molle Institute for Artificial Intelligence, Manno, Switzerland

Abstract. Deep learning with neural networks is applied by an increasing number of people outside of classic research environments, due to the vast success of the methodology on a wide range of machine perception tasks. While this interest is fueled by beautiful success stories, practical work in deep learning on novel tasks without existing baselines remains challenging. This paper explores the specific challenges arising in the realm of real world tasks, based on case studies from research & development in conjunction with industry, and extracts lessons learned from them. It thus fills a gap between the publication of latest algorithmic and methodical developments, and the usually omitted nitty-gritty of how to make them work. Specifically, we give insight deep learning projects on face matching, print media monitoring, industrial quality control, music scanning, strategy game playing, and automated machine learning, thereby providing best practices for deep learning in practice.

Keywords: data availability · deployment · loss & reward shaping · real world tasks

1 Introduction

Measured for example by the interest and participation of industry at the annual NIPS conference[1], it is save to say that deep learning [52] has successfully transitioned from pure research to application [34]. Major research challenges still exist, e.g. in the areas of model interpretability [41] and robustness [1], or general understanding [56] and stability [71,27] of the learning process, to name a few. Yet, and in addition, another challenge is quickly becoming relevant: in the light of more than 180 deep learning publications per day in the last year[2], the growing number of deep learning engineers as well as prospective researchers in the field needs to get educated on best practices and what works and what doesn't *"in the wild"*. This information is usually underrepresented in publications of a field that is very competitive and thus striving

[1] See https://medium.com/syncedreview/a-statistical-tour-of-nips-2017-438201fb6c8a.
[2] Google scholar counts > 68,000 articles for the year 2017 as of June 11, 2018.

above all for novelty and benchmark-beating results [40]. Adding to this fact, with a notable exception [22], the field lacks authoritative and detailed textbooks by leading representatives. Learners are thus left with preprints [39,61], cookbooks [46], code[3] and older gems [31,30,62] to find much needed practical advice.

In this paper, we contribute to closing this gap between cutting edge research and application in the wild by presenting case-based best practices. Based on a number of successful industry-academic research & development collaborations, we report what specifically enabled success in each case. The presented lessons learned (a) come from real-world and business case-backed use cases beyond purely academic competitions; (b) go deliberately beyond what is usually reported in our research papers in terms of tips & tricks, thus complementing them by the stories behind the scenes; (c) include also what didn't work despite contrary intuition; and (d) have been selected to be transferable to other use cases and application domains.

We organize the main part of this paper by case studies to tell the story behind each undertaking. Per case, we briefly introduce the application as well as the specific (research) challenge behind it; sketch the solution (referring details to elsewhere, as the final model architecture etc. is not the focus of this work); highlight what measures beyond textbook knowledge and published results where necessary to arrive at the solution; and show, wherever possible, examples of the arising difficulties to exemplify the challenges. Section 2 introduces a *face matching* application and the amount of surrounding models needed to make it practically applicable. Likewise, Section 3 describes the additional amount of work to deploy a state-of-the-art machine learning system into the wider IT system landscape of an *automated print media monitoring* application. Section 4 discusses interpretability and class imbalance issues when applying deep learning for *images-based industrial quality control*. In Section 5, measures to cope with the instability of the training process of a complex model architecture for large-scale *optical music recognition* are presented, and the class imbalance problem has a second appearance. Section 6 reports on practical ways for deep reinforcement learning in *complex strategy game play* with huge action and state spaces in non-stationary environments. Finally, Section 7 presents first results on comparing practical *automated machine learning* systems with the scientific state of the art, hinting at the use of simple baseline experiments. Section 8 summarizes the lessons learned and gives an outlook on future work on deep learning in practice.

2 Face matching

Designing, training and testing deep learning models for application in face recognition comes with all the well known challenges like choosing the architecture, setting hyperparameters, creating a representative training/dev/test dataset, preventing bias or overfitting of the trained model, and more. Anyway, very good results have been reported in the literature [44,53,9]. Although the challenges in lab conditions are not to be taken lightly, a new set of difficulties emerges when deploying these models in a real product. Specifically, during development, it is known what to expect as input

[3] See e.g. https://modelzoo.co/.

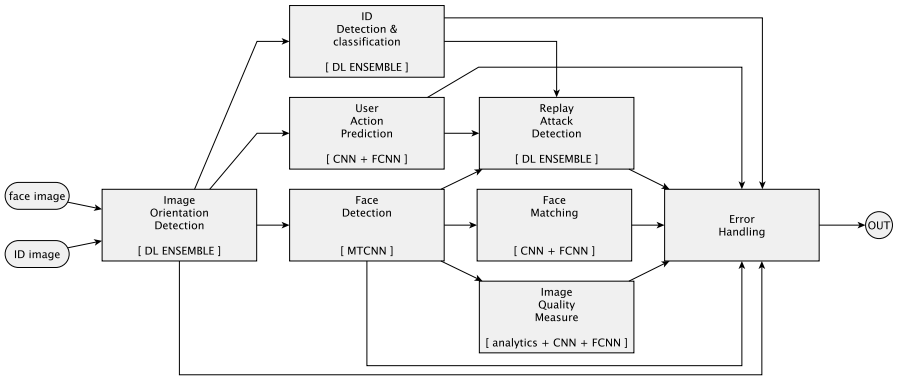

Fig. 1: Schematic representation of a face matching application with ID detection, anti-spoofing and image quality assessment. For any pair of input images (selfie and ID document), the output is the match probability and type of ID document, if no anomaly or attack has been detected. Note that all boxes contain at least one or several deep learning (DL) models.

in the controlled environment. When the models are integrated in a product that is used "in the wild", however, all kinds of input can reach the system, making it hard to maintain a consistent and reliable prediction. In this section, we report on approaches to deal with related challenges in developing an actual face-ID verification product.

Although the core functionality of such a product is to quantify the match between a person's face and the photo on the given ID, more functionality is needed to make the system perform its task well, most of it hidden from the user. Thus, in addition to the actual face matching module, the final system contains at least the following machine learnable modules (see Figure 1):

Image orientation detection When a user takes a photo of the ID on a flat surface using a mobile phone, in many cases the image orientation is random. A deep learning method is applied to predict the orientation angle, used to rotate the image in the correct orientation.

Image quality assessment consists of an ensemble of analytical functions and deep learning models to test if the photo quality is sufficient for a reliable match. It also guides the user to improve the picture taking process in case of bad quality.

User action prediction uses deep learning to predict the action performed by the user to guide the system's workflow, e.g. making a selfie, presenting an ID or if the user is doing something wrong during the sequence.

Anti-Spoofing is an essential module that uses various methods to detect if a person is showing his "real" face or tries to fool the system with a photo, video or mask. It consists of an ensemble of deep learning models.

For a commercial face-ID product, the anti-spoofing module is both most crucial for success, and technically most challenging; thus, the following discussion will focus on anti-spoofing in practice. Face matching and recognition systems are vulnerable to spoofing attacks made by non-real faces, because they are not per se able to detect

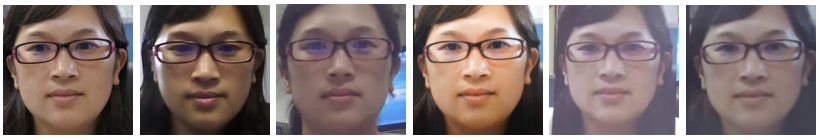

Fig. 2: Samples from the CASIA dataset [70], where photo 1, 2, and 3 on the left hand side show a real face, photo 4 shows a replay attack from a digital screen, and photos 5 and 6 show replay attacks from print.

whether or not a face is "live" or "not-live", given only a single image as input in the worst case. If control over this input is out of the system's reach e.g. for product management reasons, it is then easy to fool the face matching system by showing a photo of a face from screen or print on paper, a video or even a mask. To guard against such spoofing, a secure system needs to be able to do live-ness detection. We'd like to highlight the methods we use for this task, in order to show the additional complexity of applying face recognition in a production environment over lab conditions.

One of the key features of spoofed images is that they usually can be detected because of degraded image quality: when taking a photo of a photo, the quality deteriorates. However, with high quality cameras in modern mobile phones, looking at image quality only is not sufficient in the real world. How then can a spoof detector be designed that approves a real face from a low quality grainy underexposed photo taken by an old 640×480 web cam, and rejects a replay attack using a photo from a retina display in front of a 4K video camera (compare Figure 2)?

Most of the many spoofing detection methods proposed in the literature use hand crafted features, followed by shallow learning techniques, e.g. SVM [20,36,32]. These techniques mainly focus on texture differences between real and spoofed images, differences in color space [7], Fourier spectra [32], or optical flow maps [6]. In more recent work, deep learning methods have been introduced [3,68,67,33]. Most methods have in common that they attempt to be a one-size-fits-all solution, classifying all incoming cases with one method. This might be facilitated by the available datasets: to develop and evaluate anti-spoofing tools, amongst others CASIA [70], MSU-USSA [45], and the Replay Attack Database [12] exist. Although these datasets are challenging, they turn out to be too easy compared to the input in a production environment.

The main differences between real cases and training examples from these benchmark databases are that the latter ones have been created with a low variety of hardware devices and only using few different locations and light conditions. Moreover, the quality of images throughout the training sets is quite consistent, which does not reflect real input. In contrast, the images that the system receives "in the wild" have the most wide range of possible used hardware and environmental conditions, making the anticipation of new cases difficult. Designing a single system that can classify all such cases with high accuracy seems therefore unrealistic.

We therefore create an ensemble of experts, forming a final verdict from 3 independent predictions: the first method consists of 2 patch-based CNNs, one for low resolution images, the other one for high resolution images. They operate on fixed-size tiles from the unscaled input image using a sliding window. This technique proves to be effective for low and high quality input. The second method uses over

(a) (b) (c)

Fig. 3: Good (a) and bad (b) segmentations (blue lines denote crop marks) for realistic pages, depending on the freedom in the layout. Image (c) shows a non-article page that is excluded from automatic segmentation.

20 image quality measures as features combined with a classifier. This method is still very effective when the input quality is low. The third method uses a RNN with LSTM cells to conduct a joint prediction over multiple frames (if available). It is effective in discriminating micro movements of a real face against (simple) translations and rotations of a fake face, e.g. from a photo on paper or screen. All methods return a real vs. fake probability. The outputs of all 3 methods are fed as input features to the final decision tree classifier. This ensemble of deep learning models is experimentally determined to be much more accurate than using any known method individually.

Note that as attackers are inventive and come up with new ways to fool the system quickly, it is important to update the models with new data quickly and regularly.

3 Print media monitoring

Content-based print media monitoring serves the task of delivering cropped digital articles from printed newspapers to customers based on their pre-formulated information need (e.g., articles about their own coverage in the media). For this form of article-based information retrieval, it is necessary to segment tens of thousands of newspaper pages into articles daily. We successfully developed neural network-based models to learn how to segment pages into their constituting pages and described their details elsewhere [61,37] (see example results in Figure 3). In this section, we present challenges faced and learnings gained from integrating a respective model into a production environment with strict performance and reliability requirements.

Exclusion of non-article pages A common problem in print segmentation are special pages that contain content that doesn't represent articles in the common sense, for example classified ads, reader's letters, TV program, share prices, or sports results (see Figure 3c). Segmentation rules for such pages can be complicated, subjective, and provide little value for general use cases. We thus utilize a random forest-based

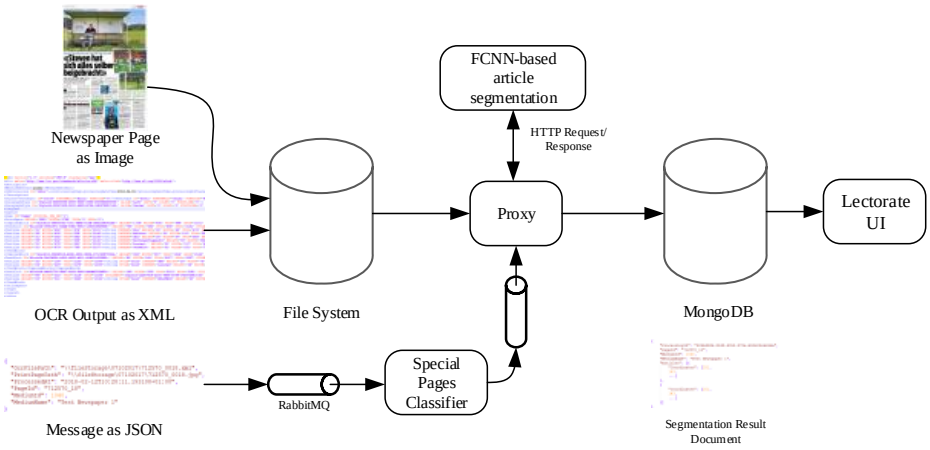

Fig. 4: Architecture of the overall pipeline: the actual model is encapsulated in the "FCNN-based article segmentation" block. Several other systems are required to warrant full functionality: (a) the *Proxy* is responsible to control data input and output from the segmentation model; (b) *RabbitMQ* controls the workflow as a message broker; (c) *MongoDB* stores all segmentation results and metrics; (d) the *Lectorate* UI visualizes results for human assessment and is used to create training data.

classifier on handcrafted features to detect such content and avoid feeding respective pages to the general segmentation system to save compute time.

Model management One advantage of an existing manual segmentation pipeline is the abundance of high quality, labeled training data being produced daily. To utilize this constant flow of data, we have started implementing an on-line learning system [55] where results of the automatic segmentation can be corrected within the regular workflow of the segmentation process and fed back to the system as training data.

After training, an important business decision is the final configuration of a model, e.g. determining a good threshold for cuts to weigh between precision and recall, or the decision on how many different models should be used for the production system. We determined experimentally that it is more effective to train different models for different publishers: the same publisher often uses a similar layout, even for different newspapers and magazines, while differences between publishers are considerable. To simplify the management of these different models, they are decoupled from the code. This is helpful for rapid development and experimentation.

Technological integration For smooth development and operation of the neural network application we have chosen to use a containerized microservices architecture [14] utilizing Docker [66] and RabbitMQ [28]. This decoupled architecture (see Figure 4) brings several benefits especially for machine learning applications: (a) a *separation of concerns* between research, ops and engineering tasks; (b) *decoupling of models/data from code*, allowing for rapid experimentation and high flexibility when deploying the individual components of the system. This is further improved by a modern devops pipeline consisting of continuous integration (CI), continuous

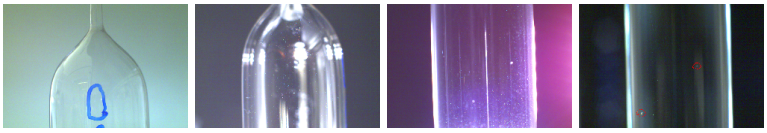

Fig. 5: Balloon catheter images taken under different optical conditions, exposing (left to right) high reflections, low defect visibility, strong artifacts, and a good setup.

deployment (CD), and automated testing; (c) *infrastructure flexibility*, as the entire pipeline can be deployed to an on-premise data-center or in the cloud with little effort. Furthermore, the use of Nvidia-docker [66] allows to utilize GPU-computing easily on any infrastructure; (d) precise *controlling and monitoring* of every component in the system is made easy by data streams that enable the injection and extraction of data such as streaming event arguments, log files, and metrics at any stage of the pipeline; and (e) easy *scaling* of the various components to fit different use cases (e.g. training, testing, experimenting, production). Every scenario requires a certain configuration of the system for optimal performance and resource utilization.

4 Visual quality control

Manual inspection of medical products for in-body use like balloon catheters is time-consuming, tiring and thus error-prone. A semi-automatic solution with high precision is thus sought. In this section, we present a case study of deep learning for visual quality control of industrial products. While this seems to be a standard use case for a CNN-based approach, the task differs in several interesting respects from standard image classification settings:

Data collection and labeling are one the most critical issues in most practical applications. Detectable defects in our case appear as small anomalies on the surface of transparent balloon catheters, such as scratches, inclusions or bubbles. Recognizing such defects on a thin, transparent and reflecting plastic surface is visually challenging even for expert operators that sometimes refer to a microscope to manually identify the defects. Thus, approx. 50% of a multi-year project time was used on finding and verifying the optimal optical settings for image acquisition. Figure 5 depicts the results of different optical configurations for such photo shootings. Finally, operators have to be trained to produce consistent labels usable for a machine learning system. In our experience, the labeling quality rises if all involved parties have a basic understanding of the methods. This helps considerably to avoid errors like e.g. only to label a defect on the first image of a series of shots while rotating a balloon: while this is perfectly reasonable from a human perspective (once spotted, the human easily tracks the defect while the balloon moves), it is a no-go for the episodic application of a CNN.

Network and training design for practical applications experiences challenges such as class imbalance, small data regimes, and use case-specific learning targets apart from standard classification settings, making non-standard loss functions necessary (see also Section 5). For instance, in the current application, we are looking for relatively small defects on technical images. Therefore, architectures proposed for

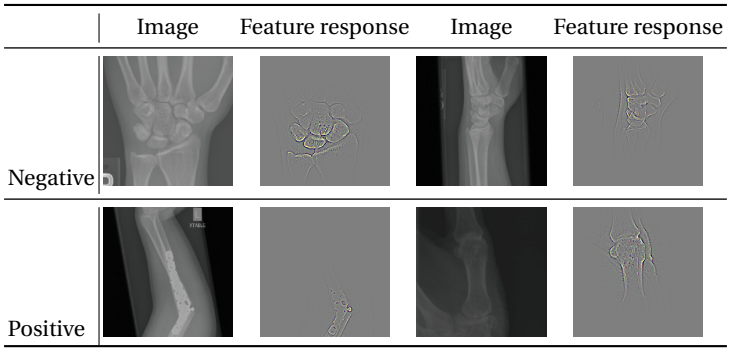

Fig. 6: Visualizing VGG19 feature responses: the first row contains two negative examples (healthy patient) and the second row positives (containing anomalies). All depicted samples are correctly classified.

large-scale image classification such as AlexNet [29], GoogLeNet [63], ResNet [26] and modern variants are not necessarily successful, and respective architectures have to be adapted to learn the relevant task. With Potential solutions for the class imbalance problem are for example:

- Down-sampling the majority class
- Up-sampling the minority class via image augmentation [13]
- Using pre-trained networks and applying transfer learning [43]
- Increasing the weight of optimization loss for the minority class [8]
- Generating synthetic data for the minority class using SMOTE [11] or GANs [23]

Selecting a suitable data augmentation approach according for the task is a necessity for its success. For instance, in the present case, axial scratches are more important than radial ones, as they can lead to a tearing of the balloon and its subsequent potentially lethal remaining in a patient's body. Thus, using 90° rotation for data augmentation could be fatal. Information like this is only gained in close collaboration with domain experts.

Interpretability of models received considerable attention recently, spurring hopes both of users for transparent decisions, and of experts on "debugging" the learning process. The latter might lead for instance to improved learning from few labeled examples through semantic understanding of the middle layers and intermediate representations in a network. Figure 6 illustrates some human-interpretable representations of the inner workings of a CNN on the recently published MUsculoskeletal RAdiographs (MURA) dataset [48] that we use here as a proxy for the balloon dataset. We used guided-backpropagation [59] and a standard VGG19 network [58] to visualize the feature responses, i.e. the part of the X-ray image on which the network focuses for its decision on "defect" (e.g., broken bone, foreign object) or "ok" (natural and healthy body part). It can be seen that the network mostly decides based on joints and detected defects, strengthening trust in its usefulness. We described elsewhere [2] that this visualization can be extended to an automatic defense against adversarial attacks [23] on deployed neural networks by thresholding the local spatial entropy

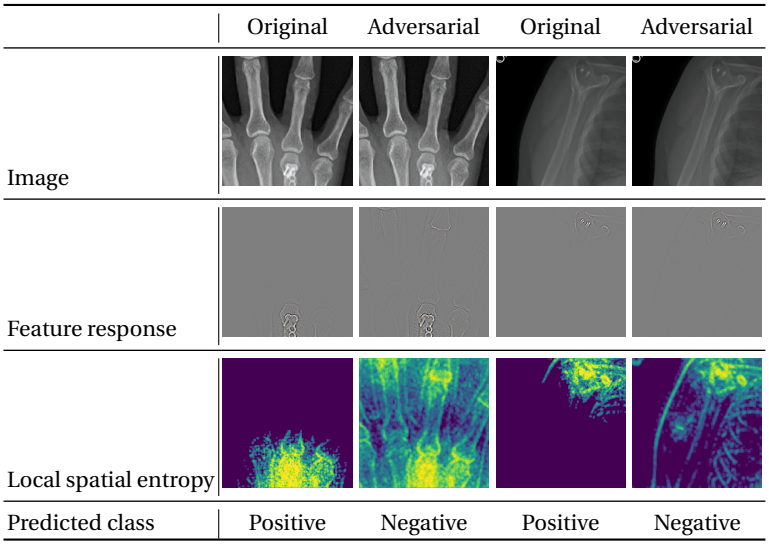

Fig. 7: Input, feature response and local spatial entropy for clean and adversarial images, respectively. We used VGG19 to estimate predictions and the Fast Gradient Sign Attack (FGSM) method [23] to compute the adversarial perturbation.

[10] of the feature response. As Figure 7 depicts, the focus of a model under attack widens considerably, suggesting that it "doesn't know where to look" anymore.

5 Music scanning

Optical music recognition (OMR) [49] is the process of translating an image of a page of sheet music into a machine-readable structured format like MusicXML. Existing products exhibit a symbol recognition error rate that is an order of magnitude too high for automatic transcription under professional standards, but don't leverage deep learning computer vision capabilities yet. In this section, we therefore report on the implementation of a deep learning approach to detect and classify all musical symbols on a full page of written music in one go, and integrate our model into the open source system Audiveris[4] for the semantic reconstruction of the music. This enables products like digital music stands based on active sheets, as most of todays music is stored in image-based PDF files or on paper.

We highlight four typical issues when applying deep learning techniques to practical OMR: (a) the absence of a comprehensive dataset; (b) the extreme data imbalance present in written music; (c) the issues of state-of-the-art object detectors with music notation (many tiny and compound symbols on large images); and (d) the transfer from synthetic data to real world examples.

Synthesizing training data The notorious data hunger of deep learning has lead to a strong dependence of results on large, well annotated datasets, such as ImageNet [51]

[4] See http://audiveris.org.

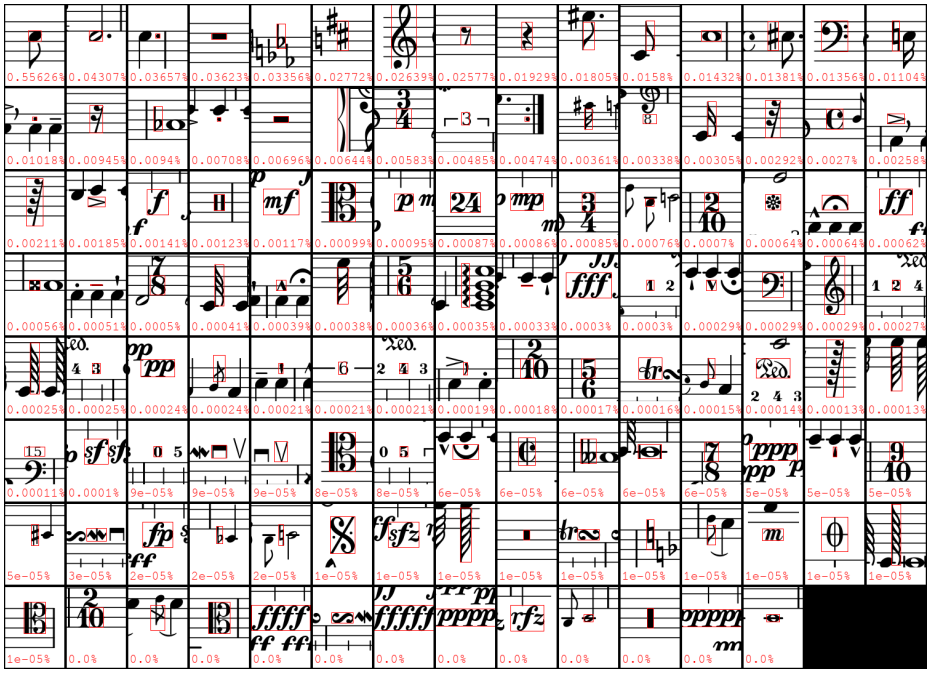

Fig. 8: Symbols classes in *DeepScores* with their relative frequencies in the dataset.

or PASCAL VOC [16]. For music object recognition, no such dataset has been readily available. Since labeling data by hand is no feasible option, we put a one-year effort in synthesizing realistic (i.e., semantically and syntactically correct music notation) data and the corresponding labeling from renderings of publicly available MusicXML files and recently open sourced the resulting *DeepScores* dataset [64].

Dealing with imbalanced data While typical academic training datasets are nicely balanced [51,16], this is rarely the case in datasets sourced from real world tasks. Music notation (and therefore *DeepScores*) shows an extreme class imbalance (see Figure 8). For example, the most common class (note head black) contains more than half of the symbols in the entire dataset, and the top 10 classes contain more than 85% of the symbols. At the other extreme, there is a class which is present only once in the entire dataset, making its detection by current machine learning methods nearly impossible. However, symbols that are rare are often of high importance in the specific pieces of music where they appear, so simply ignoring the rare symbols in the training data is not an option. A common way to address such imbalance is the use of a weighted loss function, as described in Section 4.

This is not enough in our case: first, the imbalance is so extreme that naively reweighing loss components leads to numerical instability; second, the signal of these rare symbols is so sparse that it will get lost in the noise of the stochastic gradient descent method [65], as many symbols will only be present in a tiny fraction of the mini batches. Our answer to this problem is *data synthesis* [39], using a three-fold approach to synthesize image patches with rare symbols (cp. Figure 8): (a) we

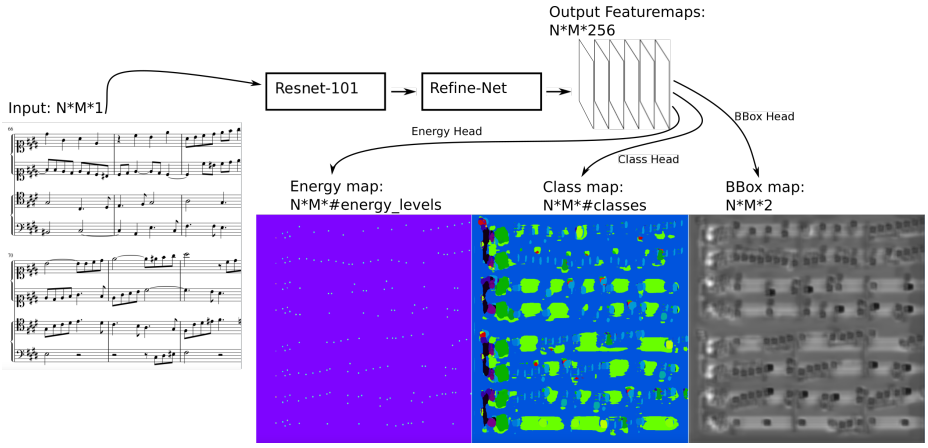

Fig. 9: Schematic of the Deep Watershed Detector model with three distinct output heads. N and M are the height and width of the input image, #classes denotes the number of symbols and #energy_levels is a hyperparameter of the system.

locate rare symbols which are present at least 300 times in the dataset, and crop the parts containing those symbols including their local context (other symbols, staff lines etc.); (b) for rarer symbols, we locate a semantically similar but more common symbol in the dataset (based on some expert-devised notion of symbol similarity), replace this common symbol with the rare symbol and add the resulting page to the dataset. This way, synthesized sheets still have semantic sense, and the network can learn from syntactically correct context symbols. We then crop patches around the rare symbols similar to the previous approach; (c) for rare symbols without similar common symbols, we automatically "compose" music containing those symbols. In all cases, we thus synthesize 300 context crops per class.

The final synthesis step is then, during training, to augment each input pages in a mini batch with 12 randomly selected synthesized crops of rare symbols (the synthesized patches have size 130×80 pixels). This way, that the neural network (on expectation) does not need to wait for more than 10 iterations to see every class which is present in the dataset. Preliminary results are promising, though more investigation is needed to see if there is any overfitting (for extremely rare symbols, the entire training is done on the 300 synthesized images), in which case different forms of preventing over-fitting should be applied (l2 regularization, additional data augmentation etc).

Enabling & stabilizing training We initially used state-of-the-art object detection models like Faster R-CNN [50] to attempt detection and classification of musical symbols on *DeepScores*. These algorithms are designed to work well on the prevalent datasets that are characterized by containing low-resolution images with a few big objects. In contrast, *DeepScores* consists of high resolution musical sheets containing hundreds of very small objects, amounting to a very different problem [64]. This disconnect lead to very poor out-of-the-box performance of said systems.

Region proposal-based systems scale badly with the number of objects present on a given image, by design. Hence, we designed the *Deep Watershed Detector* as

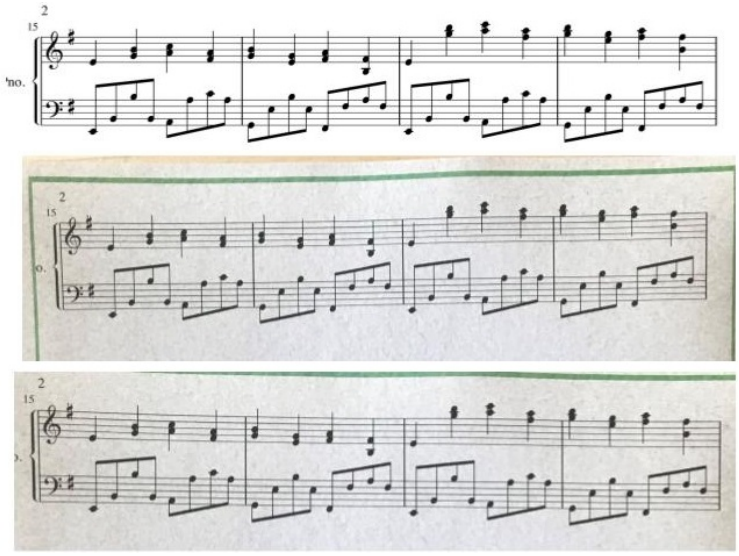

Fig. 10: Top: part of a synthesized image from *DeepScores*; middle: the same part, printed on old paper and photographed using a cell phone; bottom: the same image, automatically retrofitted to the original image coordinates for ground truth matching.

an entirely new object detection system based on the deep watershed transform [4] and described in detail elsewhere [65]. It detects raw musical symbols (e.g., not a compound note, but note head, stem and flag individually) in their context with a full sheet music page as input. As depicted in Figure 9, the underlying neural network architecture has three output heads on the last layer, each pertaining to a separate (pixel wise) task: (a) predicting the underlying symbol's class; (b) predicting the energy level (i.e., the degree of belonging of a given pixel location to an object, also called "objectness"); and (c) predicting the bounding box of the object.

Initially, the training was unstable, and we observed that the network did not learn well if it was directly trained on the combined weighted loss. Therefore, we now train the network on each of the three tasks separately. We further observed that while the network gets trained on the bounding box prediction and classification, the energy level predictions get worse. To avoid this, the network is fine-tuned only for the energy level loss after being trained on all three tasks. Finally, the network is retrained on the combined task (the sum of all three losses, normalized by their respective running means) for a few thousand iterations, giving excellent results on common symbols.

Generalizing to real-world data The basic assumption in machine learning for training and test data to be drawn from the same distribution is often violated in real world applications. In the present case, domain adaption is crucial: our training set consists of synthetic sheets created by LilyPond scripts [64], while the final product will work on scans or photographs of printed sheet music. These test pictures can have a wide variety of impairments, such as bad printer quality, torn or stained paper etc. While some work has been published on the topic of *domain transfer* [21], the results are non-satisfactory. The core idea to address this problem here is transfer learning [69]:

the neural network shall learn the core task of the full complexity of music notation from the synthetic dataset (symbols in context due to full page input), and use a much smaller dataset to adapt to the real world distributions of lighting, printing and defect.

We construct this post-training dataset by carefully choosing several hundred representative musical sheets, printing them with different types of printers on different types of paper, and finally scanning or photographing them. We then use the `BFMatcher` function from OpenCV to align these images with the original musical sheets to use all the ground truth annotation of the original musical sheet for the real-world images (see Figure 10). This way, we get annotated real-looking images "for free" that have much closer statistics to real-world images than images from *DeepScores*.

6 Game playing

In this case study, deep reinforcement learning (DRL) is applied to an agent in a multi-player business simulation video game with steadily increasing complexity, comparable to StarCraft or SimCity. The agent is expected to compete with human players in this environment, i.e. to continuously adapt its strategy to challenge evolving opponents. Thus, the agent is required to mimic somewhat general intelligent behavior by transferring knowledge to increasingly complex environments and adapting its behavior and strategies in a non-stationary, multi-agent environment with large action and state spaces. DRL is a general paradigm, theoretically able to learn any complex task in (almost) any environment. In this section, we share our experiences with applying DRL to the above described competitive environment. Specifically, the performance of a value-based algorithm using Deep Q-Networks (DQN) [38] is compared to a policy gradient method called PPO [54].

Dealing with non-stationarity In recent years, astounding results have been achieved by applying DRL in gaming environments. Examples are Atari games [38] and AlphaGo [57], where agents learned human or superhuman performance purely from scratch. In both examples, the environments were either stationary or, if an opponent was present, it did not act simultaneously in the environment. Instead, actions where taken in turns. In our environment, multiple players act simultaneously, making changes to the environment that can not be explained solely based on changes in the agents own policy. Thus, the environment is perceived as non-stationary from the agent's perspective, resulting in stability issues in RL [35]. Another source of complexity in our setting is a huge action and state space (see below). In our experiments, we observed that DQN got problems learning successful control policies as soon as the environment became more complex, even without non-stationarity induced by opponents. On the other hand, PPO's performance is generally less sensitive to non-stationarity. Further evaluation is subject of ongoing work.

Reward shaping An obvious rewarding choice is the current score of the game (or its gain). Yet, in the given environment, scoring and thus any reward based on it is sparse since it is dependent on a long sequence of correct actions on the operational, tactical and strategic level. As any rollout of the agent without scoring is not contributing to any gain in knowledge, the learning curve is flat initially. To avoid this initial phase of

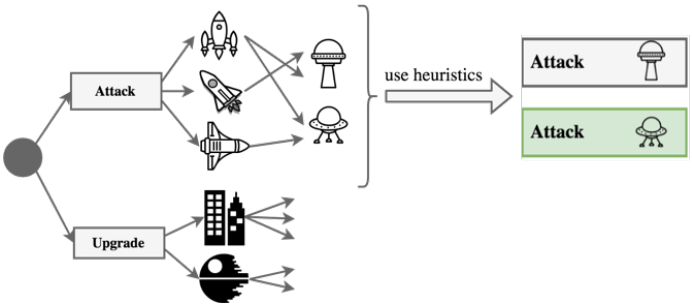

Fig. 11: Heuristic encoding of actions to prevent combinatorial explosion.

no information gain, intermediate rewards are given to individual actions, leading to faster learning progress in both DQN and PPO.

Additionally, it is not sufficient for the agent to eventually find a control policy, but it is crucial to find a good policy *quickly*, as training times are anyhow very long. Usually, comparable agents for learning complex behaviors in competitive environments are trained using self-play [5], i.e., the agents are always trained with "equally good" competitors to be be able to succeed eventually. In our setting, self play is not a straightforward first option, for several reasons: first, to jump-start learning, it is easier in our setting to play without an opponent first and only learn the art of competition later when a stable ability to act is reached; second, different from other settings, our agents should be entertaining to human opponents, not necessarily winning. It is thus not desirable to learn completely new strategies that are successful yet frustrating to human opponents. Therefore, we will investigate self-play only after stable initializations from (scripted) human opponents on different levels.

Complex state and action spaces Taking the screen frame (i.e., pixels) as input to the control policy is not applicable in our case. First, the policy's input needs to be independent of rendering and thus of hardware, game settings, game version etc. Furthermore, a current frame does not satisfy the Markov property, since attributes like "I posses item x" are not necessarily visible in it. Instead, some attributes need to be concluded from past experiences. Thus, the state space needs to be encoded into sufficient features, a task we approach with manual pre-engineering.

Next, a post-engineering approach helps in decreasing the learning time in case of DQN by removing unnecessary actions from consideration as follows: in principal, RL algorithms explore any theoretically possible state-action pair in the environment, i.e., any mathematically possible decision in the Markov Decision Process (MDP). In our environment, the available actions are dependent on the currently available in-game resources of the player, i.e., on the current state. Thus, exploring currently impossible regions in the action space is not efficient and is thus prevented by a post-engineered decision logic built to block these actions from being selected. This reduces the size of the action space per time stamp considerably. Despite changing the "learning dynamics" of the RL algorithm by externally manipulating the MDP this way, these rules where crucial in producing first satisfying learning results in our environment using DQN in a stationary setting of the game. However, when training the agent with PPO, hand-engineered rules where not necessary for proper learning.

The major problem however is the huge action and state space, as it leads to ever longer training times and thus long development cycles. It results from the fact that one single action in our environment might consist of a sequence of sub-decisions. Think e.g. of an action called “attack” in the game of StarCraft, answering the question of WHAT to do (see Figure 11). It is incompletely defined as long as it does not state WHICH opponent is to be attack using WHICH unit. In other words, each action itself requires a number of different decisions, chosen from different subcategories. To avoid the combinatorial explosion of all possible completely defined actions, we perform another post-processing on the resource management: WHICH unit to choose on WHICH type of enemy, for example, is hard-coded into heuristic rules.

This case study is work in progress, but what becomes evident already is that the combination of the complexity of the task (i.e., acting simultaneously on the operational, tactical and strategic level with exponentially increasing time horizons, as well as a huge state and action space) and the non-stationary environment prevent successful end-to-end learning as in "Pong from pixels"[5]. Rather, it takes manual pre- and post-engineering to arrive at a first agent that learns, and it does so better with policy-based rather than DQN-based algorithms. A next step will explore an explicitly hierarchical learner to cope with the combinatorial explosion of the the action space on the three time scales without using hard-coded rules, but instead factorizing the action space into subcategories.

7 Automated machine learning

One of the challenging tasks in applying machine learning successfully is to select a suitable algorithm and set of hyperparameters for a given dataset. Recent research in automated machine learning [18,42] and respective academic challenges [24] accurately aimed at finding a solution to this problem for sets of practically relevant use cases. The respective Combined Algorithm Selection and Hyperparameter (CASH) optimization problem is defined as finding the best algorithm A^* and set of hyperparameters λ_* with respect to an arbitrary cross-validation loss $\mathscr{L}$ as follows:

$$A^*, \lambda_* \in \underset{A^{(j)} \in \mathscr{A}, \lambda \in \Lambda^{(j)}}{\arg\min} \frac{1}{K} \sum_{i=1}^{K} \mathscr{L}(A_\lambda^{(j)}, D_{train}^{(i)}, D_{valid}^{(i)})$$

where $\mathscr{A}$ is a set of algorithms, Λ the set of hyperparameters per algorithm (together they form the hypothesis space), K is the number of cross validation folds and D are datasets. In this section, we compare two methods from the scientific state-of-the-art (one uses Bayesian optimization, the other genetic programming) with a commercial automated machine learning prototype based on random search.

Scientific state-of-the-art Auto-sklearn [18] is the most successful automated machine learning framework in past competitions [25]. The algorithm starts with extracting meta-features from the given dataset and finds models which perform well on similar datasets (according to the meta-features) in a fixed pool of stored successful

[5] Compare http://karpathy.github.io/2016/05/31/rl/.

Dataset	Task	Metric	Auto-Sklearn		TPOT		DSM	
			Validation	Test	Validation	Test	Validation	Test
Cadata	Regression	Coefficient Of Determination	0.7831	0.7579	**0.7930**	**0.7852**	0.7078	0.7119
Christine	Binary Classification	Balanced Accuracy Score	0.7288	0.7442	0.7269	**0.7503**	**0.7362**	0.7146
Digits	Multiclass Classification	Balanced Accuracy Score	**0.9427**	0.9378	0.9400	**0.9396**	0.8900	0.8751
Fabert	Multiclass Classification	Accuracy Score	**0.7148**	**0.7217**	0.7075	0.6950	0.7112	0.6942
Helena	Multiclass Classification	Balanced Accuracy Score	**0.3057**	**0.3125**	0.2833	0.2908	0.2085	0.2103
Jasmine	Binary Classification	Balanced Accuracy Score	0.8087	0.8304	**0.8154**	0.8147	0.8020	**0.8371**
Madeline	Binary Classification	Balanced Accuracy Score	0.8599	0.8514	**0.8949**	**0.8747**	0.7707	0.7686
Philippine	Binary Classification	Balanced Accuracy Score	0.7667	0.7554	**0.7839**	**0.7646**	0.7581	0.7406
Sylvine	Binary Classification	Balanced Accuracy Score	0.9375	0.9428	**0.9414**	**0.9480**	**0.9414**	0.9233
Volkert	Multiclass Classification	Accuracy Score	0.6659	**0.6646**	**0.6738**	0.6574	0.5220	0.5153
Average Performance			0.7514	0.7518	**0.7560**	**0.7520**	0.7048	0.6991

Table 1: Comparison of different automated machine learning algorithms.

machine learning endeavors. Auto-sklearn then performs meta-learning by initializing a set of model candidates with the model and hyperparameters choices of k nearest neighbors in dataset space; subsequently, it optimizes their hyperparameters and feature preprocessing pipeline using Bayesian optimization. Finally, an ensemble of the optimized models is build using a greedy search. On the other side, Tree-based Pipeline Optimization Tool (TPOT) [42] is toolbox based on genetic programming. The algorithm starts with random initial configurations including feature preprocessing, feature selection and a supervised classifier. At every step, the top 20% best models are retained and randomly modified to generate offspring. The offspring competes with the parent, and winning models proceed to the iteration of the algorithm.

Commercial prototype The Data Science Machine (DSM) is currently used inhouse for data science projects by a business partner. It currently uses random sampling of the solution space for optimization. Machine learning algorithms in this system are leveraged from Microsoft Azure, scikit-learn and can be user-enhanced. DSM can be deployed in the cloud, on-premise, as well as standalone. The pipeline of DSM includes data preparation, feature reduction, automatic model optimization, evaluation and final ensemble creation. The question is: can it prevail against much more sophisticated systems even at this early stage of development?

Evaluation is performed using the protocol of the AutoML challenge [24] for comparability, confined to a subset of ten datasets that is processable for the current DSM prototype (i.e., non-sparse, non-big) and containing the tasks of regression, binary- and multi-class classification. For applicability, we constrain the time budget of the searches by limiting the number of models trained with each algorithm to 100. A performance comparison is given in Table 1, suggesting that Bayesian optimization and genetic programming are superior to random search. However, random parameter search lead to reasonably good models and useful results as well (also in commercial practice). This suggests room for improvement in actual meta-*learning*.

8 Conclusions

Does deep learning work in the wild, in business and industry? In the light of the presented case studies, a better questions is: *what does it take to make it work?* Apparently, the challenges are different compared to academic competitions: instead

of a given task and known (but still arbitrarily challenging) environment, given by data and evaluation metric, real-world applications are characterized by (a) data quality and quantity issues; and (b) unprecedented (thus: unclear) learning targets. This reflects the different nature of the problems: competitions provide a controlled but unexplored environment to facilitate the discovery of new methods; real-world tasks on the other hand build on the knowledge of a zoo of methods (network architectures, training methods) to solve a specific, yet still unspecified (in formal terms) task, thereby enhancing the method zoo in return in case of success. The following lessons learned can be drawn from our six case studies (section numbers given in parentheses refer to respective details):

Data acquisition usually needs much more time than expected (4), yet is the basis for all subsequent success (5). Class imbalance and covariate shift are usual (2,4,5).

Understanding of what has been learned and how decisions emerge help both the user and the developer of neural networks to build trust and improve quality (4,5). Operators and business owners need a basic understanding of used methods to produce usable ground truth and provide relevant subject matter expertise (4).

Deployment should include on-line learning (3) and might involve the buildup of up to dozens of other machine learning models (2, 3) to flank the original core part.

Loss/reward shaping is usually necessary to enable learning of very complex target functions in the first place (5,6). This includes encoding expert knowledge manually into the model architecture or training setup (4, 6), and handling special cases separately (3) using some automatic pre-classification.

Simple baselines do a good job in determining the feasibility as well as the potential of/in the task at hand when final datasets or novel methods are not yet seen (4, 7). Increasing the complexity of methods and (toy-)tasks in small increments helps monitoring progress, which is important to effectively debug failure cases (6).

Specialized models for identifiable sub-problems increase the accuracy in production systems over all-in-one solutions (2,3), and ensembles of experts help where no single method reaches adequate performance (2).

Best practices are straightforward to extract on the general level ("plan enough resources for data acquisition"), yet quickly get very specific when broken down to technicalities ("prefer policy-based RL methods"). An overarching scheme seems to be that the specific challenges in real-world tasks need similar amounts of creativity and knowledge to get solved as fundamental research tasks, suggesting they need similar development methodologies on top of proper engineering and business planning.

Future work to tame the wilderness will concentrate on the following four aspects: (a) making deep learning more sample efficient to cope with smaller training data (or at least: label) sets, e.g. by one-shot learning [17,19], data generation [47], label generation [15] or architecture learning [60]; (b) finding suitable architecture combinations and loss designs to cope with the complexity of real-world tasks; (c) improve the stability of training and prediction, and thus also (d) the robustness and interpretability of neural network models for different modalities.

Acknowledgements We are grateful for the invitation by the ANNPR chairs and the support of our business partners in Innosuisse grants 17719.1 "PANOPTES", 17963.1 "DeepScore", 25256.1 "Libra", 25335.1 "FarmAI", 25948.1 "Ada" and 26025.1 "QualitAI".

References

1. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: A survey. arXiv preprint arXiv:1801.00553 (2018)
2. Amirian, M., Schwenker, F., Stadelmann, T.: Trace and detect adversarial attacks on cnns using feature response maps. In: ANNPR (2018)
3. Atoum, Y., Liu, Y., Jourabloo, A., Liu, X.: Face anti-spoofing using patch and depth-based cnns. In: IEEE Int. Joint Conference on Biometrics (IJCB). pp. 319–328 (2017)
4. Bai, M., Urtasun, R.: Deep watershed transform for instance segmentation. In: CVPR (2017)
5. Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., Mordatch, I.: Emergent complexity via multi-agent competition. arXiv preprint arXiv:1710.03748 (2017)
6. Bao, W., Li, H., Li, N., Jiang, W.: A liveness detection method for face recognition based on optical flow field. Int. Conference on Image Analysis and Signal Processing (2009)
7. Boulkenafet, Z., Komulainen, J., Hadid, A.: Face anti-spoofing based on color texture analysis. In: Int. Conference on Image Processing (ICIP) (2015)
8. Buda, M., Maki, A., Mazurowski, M.A.: A systematic study of the class imbalance problem in convolutional neural networks. arXiv preprint arXiv:1710.05381 (2017)
9. Cao, Q., Shen, L., Xie, W., Parkhi, O.M., Zisserman, A.: Vggface2: A dataset for recognising faces across pose and age. arXiv preprint arXiv:1710.08092 (2017)
10. Chanwimaluang, T., Fan, G.: An efficient blood vessel detection algorithm for retinal images using local entropy thresholding. Int. Symposium on Circuits and Systems (ISCAS) **5** (2003)
11. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research **16**, 321–357 (2002)
12. Chingovska, I., Anjos, A., Marcel, S.: On the effectiveness of local binary patterns in face anti-spoofing. In: BIOSIG. pp. 1–7 (2012)
13. Ciresan, D.C., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: CVPR. pp. 3642–3649 (2012)
14. Dragoni, N., Lanese, I., Larsen, S.T., Mazzara, M., Mustafin, R., Safina, L.: Microservices: How to make your application scale. In: International Andrei Ershov Memorial Conference on Perspectives of System Informatics. pp. 95–104. Springer (2017)
15. Elezi, I., Torcinovich, A., Vascon, S., Pelillo, M.: Transductive label augmentation for improved deep network learning. In: ICPR (2018)
16. Everingham, M., Gool, L.J.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes (VOC) challenge. Int. Journal of Computer Vision **88**(2), 303–338 (2010)
17. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Trans. PAMI **28**(4), 594–611 (2006)
18. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: NIPS. pp. 2962–2970 (2015)
19. Finn, C., Yu, T., Zhang, T., Abbeel, P., Levine, S.: One-shot visual imitation learning via meta-learning. In: Conference on Robot Learning (CoRL) (2017)
20. Galbally, J., Marcel, S., Fiérrez, J.: Image quality assessment for fake biometric detection: Application to iris, fingerprint, and face recognition. IEEE Trans. Image Processing **23**(2), 710–724 (2014)
21. Gebru, T., Hoffman, J., Fei-Fei, L.: Fine-grained recognition in the wild: A multi-task domain adaptation approach. In: ICCV (2017)
22. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
23. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. ICLR (2015)
24. Guyon, I., Bennett, K., Cawley, G., Escalante, H.J., Escalera, S., Ho, T.K., Macià, N., Ray, B., Saeed, M., Statnikov, A., Viegas, E.: Design of the 2015 ChaLearn AutoML challenge. In: IJCNN (2015)

25. Guyon, I., Chaabane, I., Escalante, H.J., Escalera, S., Jajetic, D., Lloyd, J.R., Macía, N., Ray, B., Romaszko, L., Sebag, M., Statnikov, A., Treguer, S., Viegas, E.: A brief review of the ChaLearn AutoML challenge. In: AutoML workshop@ICML (2016)
26. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
27. Irpan, A.: Deep reinforcement learning doesn't work yet. Online (Feb. 14): https://www.alexirpan.com/2018/02/14/rl-hard.html (2018)
28. John, V., Liu, X.: A survey of distributed message broker queues. arXiv preprint arXiv:1704.00411 (2017)
29. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. pp. 1097–1105 (2012)
30. Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring strategies for training deep neural networks. JMLR (1), 1–40 (1 2009)
31. LeCun, Y., Bottou, L., Orr, G.B., Müller, K.R.: Efficient backprop. In: Orr, G.B., Müller, K.R. (eds.) Neural networks: Tricks of the trade, pp. 9–50. Springer, Berlin, Heidelberg (1998)
32. Li, J., Wang, Y., Tan, T., Jain, A.K.: Live face detection based on the analysis of fourier spectra. Biometric Technology for Human Identification (2004)
33. Li, L., Feng, X., Boulkenafet, Z., Xia, Z., Li, M., Hadid, A.: An original face anti-spoofing approach using partial convolutional neural network. In: Int. Conference on Image Processing Theory, Tools and Applications (IPTA) (2016)
34. Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., Alsaadi, F.E.: A survey of deep neural network architectures and their applications. Neurocomputing **234**, 11 – 26 (2017)
35. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O.P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. In: NIPS. pp. 6382–6393 (2017)
36. Määttä, J., Hadid, A., Pietikäinen, M.: Face spoofing detection from single images using micro-texture analysis. In: Int. Joint Conference on Biometrics (IJCB) (2011)
37. Meier, B., Stadelmann, T., Stampfli, J., Arnold, M., Cieliebak, M.: Fully convolutional neural networks for newspaper article segmentation. In: ICDAR (2017)
38. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
39. Ng, A.: Machine Learning Yearning - Technical Strategy for AI Engineers in the Era of Deep Learning (2018), [to appear]
40. Olah, C., Carter, S.: Research debt. Distill (2017)
41. Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., Mordvintsev, A.: The building blocks of interpretability. Distill (2018)
42. Olson, R.S., Urbanowicz, R.J., Andrews, P.C., Lavender, N.A., Kidd, L.C., Moore, J.H.: Automating biomedical data science through tree-based pipeline optimization. In: European Conference Applications of Evolutionary Computation (EvoApplications). pp. 123–137 (2016)
43. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. knowledge and data engineering **22**(10), 1345–1359 (2010)
44. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. In: BMVC. pp. 41.1–41.12 (2015)
45. Patel, K., Han, H., Jain, A.K.: Secure face unlock: Spoof detection on smartphones. IEEE Trans. Information Forensics and Security **11**(10), 2268–2283 (2016)
46. Perez, C.E.: The Deep Learning AI Playbook - Strategy for Disruptive Artificial Intelligence (2017)
47. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks (2016)

48. Rajpurkar, P., Irvin, J., Bagul, A., Ding, D., Duan, T., Mehta, H., Yang, B., Zhu, K., Laird, D., Ball, R.L., et al.: Mura dataset: Towards radiologist-level abnormality detection in musculoskeletal radiographs. arXiv preprint arXiv:1712.06957 (2017)
49. Rebelo, A., Fujinaga, I., Paszkiewicz, F., Marçal, A.R.S., Guedes, C., Cardoso, J.S.: Optical music recognition: state-of-the-art and open issues. Int. Journal of Multimedia Information Retrieval **1**(3), 173–190 (2012)
50. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS (2015)
51. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. Int. Journal of Computer Vision **115**(3), 211–252 (2015)
52. Schmidhuber, J.: Deep learning in neural networks: An overview. Neural networks **61**, 85–117 (2015)
53. Schroff, F., Kalenichenko, D., Philbin, J.: In: CVPR (2015)
54. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
55. Shalev-Shwartz, S.: Online learning and online convex optimization. Foundations and Trends® in Machine Learning **4**(2), 107–194 (2012)
56. Shwartz-Ziv, R., Tishby, N.: Opening the black box of deep neural networks via information. arXiv preprint arXiv:1703.00810 (2017)
57. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. nature **529**(7587), 484–489 (2016)
58. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. ICLR (2015)
59. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806 (2014)
60. Srivastava, R.K., Greff, K., Schmidhuber, J.: Training very deep networks. In: NIPS (2015)
61. Stadelmann, T., Tolkachev, V., Sick, B., Stampfli, J., Dürr, O.: Beyond imagenet - deep learning in industrial practice. In: Braschler, M., Stadelmann, T., Stockinger, K. (eds.) Applied Data Science - Lessons Learned for the Data-Driven Business. Springer (2018), to appear
62. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: ICML (2013)
63. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., et al.: Going deeper with convolutions. CVPR (2015)
64. Tuggener, L., Elezi, I., Schmidhuber, J., Pelillo, M., Stadelmann, T.: Deepscores - a dataset for segmentation, detection and classification of tiny objects. In: ICPR (2018)
65. Tuggener, L., Elezi, I., Schmidhuber, J., Stadelmann, T.: Deep watershed detector for music object recognition. In: ISMIR (2018)
66. Xu, P., Shi, S., Chu, X.: Performance evaluation of deep learning tools in docker containers. arXiv preprint arXiv:1711.03386 (2017)
67. Xu, Z., Li, S., Deng, W.: Learning temporal features using LSTM-CNN architecture for face anti-spoofing. In: ACPR. pp. 141–145 (2015)
68. Yang, J., Lei, Z., Li, S.Z.: Learn convolutional neural network for face anti-spoofing. arXiv preprint arXiv:1408.5601 (2014)
69. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: NIPS. pp. 3320–3328 (2014)
70. Zhang, Z., Yan, J., Liu, S., Lei, Z., Yi, D., Li, S.Z.: A face antispoofing database with diverse attacks. In: Int. Conference on Biometrics (ICB). pp. 26–31 (2012)
71. Zheng, S., Song, Y., Leung, T., Goodfellow, I.: Improving the robustness of deep neural networks via stability training. In: CVPR. pp. 4480–4488 (2016)