Introduction

For this Machine Learning challenge, you are going to classify animals from images. There are 12 different animals, and your goal is to build a classifier that is able to distinguish which animal is depicted in an image.

Nowadays, a problem like this is almost always solved by using Convolutional Neural Networks (CNNs) directly on the RGB images, and many state-of-the-art implementations exist which classify images as well as humans do, or even better. While you can definitely try the state-of-the-art (we will set up a separate competition for deep learning entries), we will here focus on a different route using a more "old-school" approach of Visual-Bag-of-Words models. We extract meaningful vector features from images using a technique similar to the Bag-of-Words (BoW) from Natural Language Processing. This enables us to reduce an image -- with its high-dimensionality from its many RGB pixels -- to a smaller vector that contains the relevant information from that image. This vector can then be used as input to any of our classification algorithms, just like we're used to.

Prizes

When the competition is over, the participants who hold the top three positions on the final leaderboard (based on the private part of the test set) each win a copy of the book 'Human Compatible: AI and the Problem of Control', by Stuart Russell.

How to start?

You are encouraged to not only use the algorithms seen in the course (although they will provide a solid starting point), but to go beyond this and try out the many other classifiers that are out there. After all, some algorithms perform better on certain tasks than others, and it is up to you to find out which ones are best suited for this particular problem (and why!). You are expected to build a solid machine learning pipeline in which you can rigorously test the different algorithms. To do this, you will need to adhere to the following steps:

1) Problem and data analysis
2) Preprocessing and feature extraction
3) Training/validation/test split
4) Model training and hyperparameter tuning
5) Analysis of performance and errors
6) Learn from analysis and repeat

We advise you to start by checking out the 'startkit' folder, and start building your pipeline from there. Start simple, for example by trying out the Naive Bayes classifier from scikit-learn, or the Logistic Regression classifier from the attached classificiation tutorial, or a (linear) Support Vector Machine (SVM), etc. as a baseline and see how your performance improves compared to the prior probablities "algorithm" we provided in the start_here notebook. Build a strong train/test and (cross-)validation scheme around your classifiers to improve the generalization of your model, and really think about why you are doing this or that. Start checking out everything scikit-learn has to offer (feature extraction, feature selection, other classifiers, performance metrics etc.). If you are not satisfied with what scikit-learn has to offer, build your own solutions (once you progress in the challenge, this might become useful for certain parts).

Inspect your data thoroughly. Analyze every last bit of information out of your results. Are you using the right performance metrics? Are there outliers? Is your dataset balanced? If not, how do you address this (think about your performance metrics, loss function and the distributions of the train/validation/test splits)? Do you need normalization? If so, are you using the correct normalization? What are the limitations of the dataset? Can you augment your data? Do you need dimensionality reduction? What about feature selection? Do you use regularization? How did you choose your hyperparameters? Are you overfitting? Underfitting? Would having more data benefit your model? Can you quantify that with a learning curve? Is there information-leakage from your training set to your test/validation set? Are your features informative enough? Is your model powerful enough? Should you try an ensemble of models? Do you pick a linear model, or a non-linear one? Why? How could you improve the model's weak spots? Why is your model misclassifying these samples? Etcetera, etcetera.

Second competition for deep learning

You will mostly be graded on your work with the Visual-Bag-of-Words approach, but if you feel like you've exhausted what this approach (and scikit-learn) has to offer, we definitely encourage you to try and step up your game by trying out convolutional neural networks (especially for master students taking a specialization in AI). Build a (small) CNN from scratch on the dataset; it should be possible to already improve on your best results from the VBoW approach. You can use TensorFlow or PyTorch, the two most well-known deep learning frameworks to do so. If you want to go even further, you can try a technique called *transfer learning*, in which you finetune the last layers of a state-of-the-art pretrained deep CNN to solve the problem. I would suggest you start at *tensorflow.org/tutorials/images/transfer_learning*, or from any other tutorial on the web. Do not submit solutions using neural networks here, we have set up a separate competition for neural networks. If by accident you submit a neural network solution here, notify us, and do not select it as one of your two final submissons, or you will be disqualified.

Grading

At the end of the competition, we expect from you a detailed pdf report in which you explain what you did and why. Explain what techniques you started with, what worked, how you improved them etc. Try to mainly cover the successful parts of your work, but do briefly mention what you tried and discarded as unuseful as well. Include also a brief description of the more computational aspects of your work: did you use parallellisation? A GPU? How do your algorithms compare computationally? We expect you to use lots of strong graphs and visualizations, as this is a core skill of a good data scientist and helps you convey your results better. The maximal length of your report should be about 8 pages (it's fine if you need a bit more or less). We expect at least a section on data exploration, preprocessing, model selection and tuning, your results and an analysis.

Besides your complete report at the end of the competition, we expect you to hand in a smaller intermediate report as well, so that we can track everyone's progress and give useful feedback during the competition, not only after. It will be graded and it will count for a small percentage of your total project marks. This intermediate report should have a similar structure as your final report, but it's not necessary to polish it, we just want to know what you've been doing and why, so we can see what common mistakes are being made and address them. You should at least by then have a description of one simple, successful, well-tuned model (e.g., a well-tuned logistic regression classifier) and a basic analysis of it. Don't worry if you haven't tried everything yet, you can get a perfect score here by only making sure you get the basics right. Aim for about 4 pages or so (with visualizations).

Upload your pdf and code on the GitHub classroom (more details on that later via Canvas). Your pdf name should at least contain your first name, your last name, your student number and an indication whether it's your final report or your intermediate one.

While getting a good score on the leaderboard is encouraged and most likely means you are doing well, remember that you will be graded based on the quality of your report, so explain your steps and choices clearly and discuss all the issues you have faced during this challenge. Don't wait on building your report until after the competition is done!

Deadline

The deadline for the intermediate report is Friday, December 11, 23:59.

The deadline for the competition is Friday, January 15, 23:59. The deadline for the report is Sunday, January 17, 23:59.