AUDIO-BASED MUSIC CLASSIFICATION WITH A PRETRAINED CONVOLUTIONAL NETWORK

Sander Dieleman, Philémon Brakel and Benjamin Schrauwen
Electronics and Information Systems department, Ghent University
{sander.dieleman, philemon.brakel, bschrauw}@elis.ugent.be

ABSTRACT

Recently the 'Million Song Dataset', containing audio features and metadata for one million songs, was made available. In this paper, we build a convolutional network that is then trained to perform artist recognition, genre recognition and key detection. The network is tailored to summarize the audio features over musically significant timescales. It is infeasible to train the network on all available data in a supervised fashion, so we use unsupervised pretraining to be able to harness the entire dataset: we train a convolutional deep belief network on all data, and then use the learnt parameters to initialize a convolutional multilayer perceptron with the same architecture. The MLP is then trained on a labeled subset of the data for each task. We also train the same MLP with randomly initialized weights. We find that our convolutional approach improves accuracy for the genre recognition and artist recognition tasks. Unsupervised pretraining improves convergence speed in all cases. For artist recognition it improves accuracy as well.

1. INTRODUCTION

Recently, the Laboratory for the Recognition and Organization of Speech and Audio (LabROSA) [1] of Columbia University released a large dataset of music consisting of audio features and metadata for one million songs, aptly named the 'Million Song Dataset' [4].

Because the dataset is almost completely labeled, it lends itself well for developing and testing classification methods. In this paper, we attempt to classify songs according to their genre, artist and key. To this end, we design a convolutional network that summarizes the input features over musically significant timescales.

[1] http://labrosa.ee.columbia.edu/

Developing techniques that can harness the entire dataset is quite a challenge. We use the majority of the data in an unsupervised learning phase, where the network learns to model the audio features. Due to its size, the dataset is very suitable for unsupervised learning. This is followed by a supervised training phase, where only a small task-specific subset of the dataset is used to train a discriminative model using the same network. We have investigated the gains that can be achieved by using a convolutional architecture, and the additional gains that unsupervised pretraining can offer.

This paper is structured as follows: the layout of the dataset is detailed in Section 2. An introduction to convolutional deep belief networks (DBNs) follows in Section 3. Section 4 describes the classification tasks that were used to evaluate the model. Section 5 provides an overview of our approach, and Section 6 describes our experimental setup. Results are given in Section 7.

2. DATASET

2.1 The Million Song Dataset

The Million Song Dataset is a collection of all the information that is available through The Echo Nest API [2] for one million popular songs. This means that a lot of the data was automatically derived from musical audio signals, which should be taken into account when it is used for learning. Metadata available includes artist and album information and the year of the performance. Musical information derived directly from the audio signal includes the key, the mode and the time signature. Next to this, some other derived features like "energy" and "danceability" and user-assigned tags are also available.

The audio features in the dataset were obtained by first dividing each song into so-called segments. Segment boundaries roughly correspond to onsets of notes or other musical events. For each segment, a feature vector consisting of 12 timbre and 12 chroma components was computed, as well as the maximal loudness within the segment.

The chroma features describe the pitch content of the music. Each of the 12 components corresponds to a pitch class

[2] http://the.echonest.com/

(ranging from C to B). Their values indicate the relative presence of the pitches, with the most prominent one always having a value of 1. All components lie within the interval $[0, 1]$. The timbre features are the coefficients of 12 basis functions which capture certain timbral characteristics like brightness, flatness and attack. They are unbounded and roughly centered around 0.

Unfortunately, the automated methods used to build the dataset lead to the presence of a relatively large number of duplicate tracks. When the dataset is divided into a train and a test set in a naive fashion, some examples might occur in both subsets, which is undesirable. Luckily, the authors of the dataset have published an extensive list of known duplicates. Using this list, over 78,000 tracks were removed.

2.2 Beat-aligned Features

Although the segmentation that was performed to compute the audio features has its merits, we are more interested in *beat-aligned* features such as those used in [3]. The beat is the basic unit of time in music. Chord progressions and changes in musical texture tend to occur on the beat, and seeing as it is one of our goals to encode these characteristics in higher level features, it makes sense to use beat-aligned features as a starting point.

The features from the dataset can be converted to beat-aligned features using the rhythm information that is also supplied. The segments are mapped to beats, and then the feature vectors for all segments corresponding to the same beat are averaged.

3. CONVOLUTIONAL DEEP BELIEF NETWORKS

3.1 Deep Learning

A fairly recent trend in machine learning is the use of deep architectures, with many layers of processing [1]. Traditionally, such architectures were not very popular because they were very difficult to train. In 2006, Hinton demonstrated a fast training method for *deep belief networks* (DBNs), a particular type of deep models [11]. This led to a surge in popularity of these models, establishing *deep learning* as a new area of research.

The popularity of deep architectures can be attributed at least partially to their biological plausibility; humans typically use hierarchies and abstractions to organize their thoughts and evidence of hierarchical structures has been found in the brain (e.g. in the visual cortex [1]).

Deep belief networks are probabilistic generative models, which are obtained by stacking multiple restricted Boltzmann machines (RBMs) on top of eachother.

3.2 Restricted Boltzmann Machines

A restricted Boltzmann machine is a probabilistic model consisting of a set of visible units and a set of hidden units which form a bipartite graph; there are no connections between pairs of visible units or pairs of hidden units, but every visible unit is connected to every hidden unit. They are a kind of undirected graphical model. A schematic representation is shown in Figure 1.

The visible units of an RBM correspond to the input variables of the data that is to be modelled. In image processing, each visible unit typically represents one pixel. The hidden units capture correlations between visible units and can be seen as *feature detectors*. The model learns the underlying distribution of the data by representing it in terms of features that are derived from the data itself.

Each connection has a particular weight, and each of the units can also have a bias. These trainable parameters can be learnt from data. Unfortunately, maximum likelihood learning is intractable in RBMs. Instead, the *contrastive divergence* learning rule, which is an approximation to maximum likelihood learning, can be used [9].

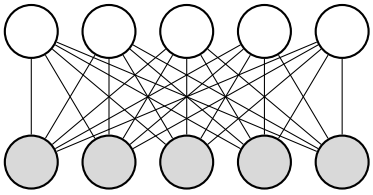

Figure 1. Schematic representation of an RBM, with the visible units at the bottom and the hidden units at the top. Note how there are no lateral connections between two visible or two hidden units.

RBMs typically consist of binary units, which can be on or off. This makes sense for the hidden units, which are feature detectors, but it is not always the best choice for the visible units. It is also possible to construct an RBM for continuous data, with Gaussian visible units.

3.3 Deep Belief Networks

A deep belief network (DBN) consists of multiple RBMs stacked on top of eachother, with the hidden units of RBM i being used as visible units of RBM $i + 1$. The bottom RBM learns a shallow model of the data. The next one then learns to model the hidden units of the first, and so on: higher-level features are extracted from lower-level features. Each RBM is trained separately; learning would be considerably harder if all layers would be trained jointly using backpropagation.

Top-level features learnt by DBNs can be used to train discriminative models. In this fashion, they have been applied succesfully to image processing problems like hand-

writing recognition [11] and object recognition [12], but also to classification of audio signals [14], and even music classification [8]. For a detailed technical overview of deep learning, RBMs and DBNs, see [1].

3.4 Convolutional Networks

A convolutional networks is a type of network model with constrained weights. There are two kinds of constraints:

- locality: each unit in layer i is only connected to a group of units in layer $i-1$ that is local;
- translation invariance: each unit in layer i is replicated such that every local group of units in layer $i-1$ is connected to a unit in layer i with the same weight configuration (*weight sharing*). A set of units in layer i with the same weight configuration is called a *feature map*.

This configuration is visualized in Figure 2.

In layered network models, we typically wish for higher layers to represent higher levels of abstraction. Weight constraints in convolutional networks would make it hard for neurons in higher layers to learn high-level abstractions; they only see a small local portion of the input, whereas high-level abstractions usually involve long-range dependencies. To increase the scope of higher layer neurons, convolutional layers are alternated with *max-pooling* layers.

Max-pooling is a downsampling operation: units in layer i are grouped into small non-overlapping blocks. Each block is aggregated into a single unit in layer $i+1$, with as its activation the maximal activation over all units in the block. This operation reduces the dimensionality of the data by a factor equal to the size of the blocks. This layout is also shown in Figure 2.

It's clear that inserting max-pooling layers between convolutional layers increases the scope of higher layer neurons. Furthermore, it also makes the model invariant to some small displacements of the input data, increasing its robustness.

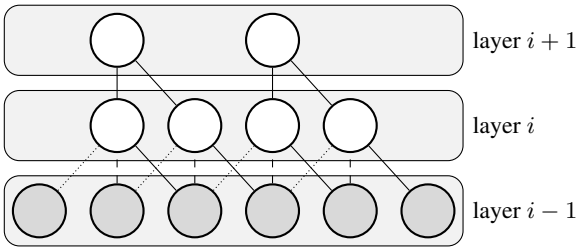

Figure 2. A max-pooling layer ($i+1$) stacked on top of a convolutional layer (i). Note that layer $i-1$ and layer i are not fully connected. The connections are drawn in different styles to indicate which weights are shared.

Convolutional networks are typically used for image processing, where stronger correlations between nearby pixels and the translation invariance of image features are exploited to significantly reduce the number of parameters. Audio signals have similar characteristics, although the locality is temporal rather than spatial.

Deep belief networks can be made convolutional by applying the described weight constraints in the RBM layers, and inserting max-pooling layers between the RBM layers. Convolutional deep belief networks have been have been used for object recognition [13, 16], and to extract features from audio signals, for speech recognition as well as for music classification [14].

3.5 Supervised Finetuning

As mentioned earlier, we can use top-level DBN features as input for a classification method; common choices are support vector machines or logistic regression. We can train a logistic regression classifier by gradient descent, using the DBN to preprocess the input data.

It is also possible to convert a DBN into a convolutional multilayer perceptron (MLP). We can simply reuse the weights of the interconnections and the biases of the hidden units. We then stack a logistic regression layer on top of this MLP and train the whole model jointly using gradient descent. This approach is called *supervised finetuning*: the DBN weights that were initially learnt to model the data are now finetuned for a specific discriminative task using backpropagation.

4. TASKS

We performed several classification tasks on music tracks: artist recognition, genre recognition and key detection. Labeled datasets for each of the tasks were extracted from the Million Song Dataset. Three (partially overlapping) subsets were selected:

- artist recognition: the 50 artists with the most tracks in the dataset were identified, and 100 tracks of each artist were selected (5000 tracks in total);
- genre recognition: 20 common genres were selected manually using tags [3] that are included in the dataset: folk, punk, metal, jazz, country, blues, classical, rnb, new wave, world, soul, latin, dance, reggae, techno, funk, rap, hip hop, rock and pop. For each genre, 250 tracks were selected (5000 tracks in total);
- key detection: the key information in the dataset was automatically annotated, so it may be unreliable. To avoid problems with incorrect labels, we selected 250 tracks with a high key confidence for each of the 12 possible keys (3000 tracks in total).

The subsets were then divided into balanced train, evaluation and test sets according to a 80% / 10% / 10% split.

[3] The dataset provides different kinds of tags. We used the MusicBrainz tags because these are the most reliable [4].

5. APPROACH

We built a convolutional network, designed to aggregate the features from the dataset on musically significant timescales. Properties that are typical for certain genres, artists or keys, should become apparent at this level. We used the same network to tackle all three classification tasks.

The network was first trained as a DBN on the entire Million Song Dataset [4] . We then trained and evaluated the network as an MLP with backpropagation, for each of the classification tasks. We used the Theano Python library to implement all experiments, so they could be GPU-accelerated easily [2].

5.1 Network Layout

The input of the network consists of beat-aligned chroma and timbre features for a given track, so there are 24 input dimensions in total. The maximal loudness component was not used, as the timbre features already include a loudness component. Note that tracks vary considerably in length, but the convolutional nature of the network allows us to cope easily with variable-length input.

First, we separated the chroma and timbre features into two input layers (layers 0a and 0b). Then, separate convolutional layers were stacked onto both input layers (layers 1a and 1b). These layers learn features with a width of 8 beats. It was observed that most of the tracks in the dataset have a 4/4 time signature (which is also true for contemporary music in general). This means that there are 4 beats in a bar. The width of the features was chosen to be two bars, seeing as this is the timescale on which chord progressions and changes in musical texture are most likely to occur. We used 100 feature maps for each layer.

By using separate layers, the network does not learn correlations between chroma and timbre features at this level. This allows it to focus on learning correlations between timbre components and between chroma components separately; such correlations are likely to be easier to discover. A similar approach was used in [15] to learn features over multiple modalities.

The output of the convolutional layers was then max-pooled in the time dimension with a pool size of 4 (layers 2a and 2b). Once again, we made use of the observation that most of the tracks in the dataset have a 4/4 time signature, with 4 beats per bar; the output of the max-pooling layer is invariant to all displacements of less than one bar (up to 3 beats).

The max-pooled outputs of both layers were then concatenated, yielding 200 features with a granularity of approximately 1 bar. We stacked another convolutional layer with 100 feature maps on top of this, which learns features with a width of 8 bars (layer 3). This width was selected because musical themes are often contained within a length of 8 bars. Correlations between timbre and chroma components can now be discovered as well.

Finally, another max-pooling layer with a pool size of 4 was added (layer 4). The features obtained from this layer have a granularity of 4 bars and a scope of roughly 8 bars. To perform the classification tasks, a fifth layer performing logistic regression was added. To classify a track, each timestep of the layer 4 is classified separately, and the resulting posterior distributions over the class labels are averaged. The most probable class is then selected. The layout of the network is shown in Figure 3.

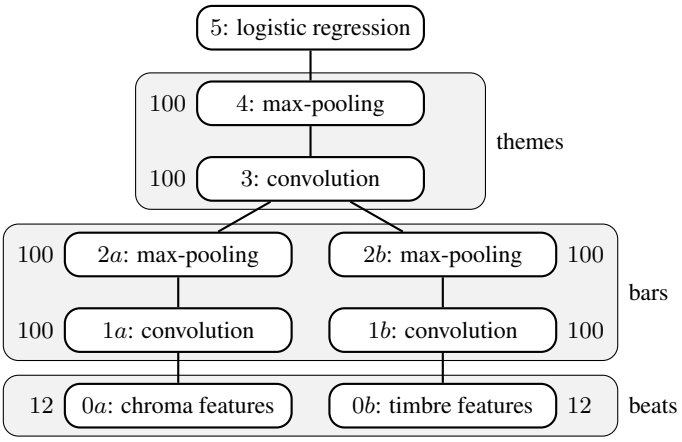

Figure 3. The network layout. The number of dimensions or feature maps for each layer is indicated on the side. The layers have also been grouped according to the timescale on which they operate.

5.2 Unsupervised Pretraining

It would be impossible to train the network in a supervised fashion with the entire Million Song Dataset. This is this computationally infeasible, and on top of that the provided labels are not perfect; some are missing, others are incorrect or have a very low confidence.

As mentioned before, we pretrained the network using timbre and chroma features for all tracks in the dataset. We used the beat-aligned chroma features directly as inputs to the network; the timbre features were first normalized per track to have zero mean and unit variance.

To train the RBM in layer 1b (timbre), we use Gaussian visible units, which allow for the continuous input data to be modeled. For layer 1a (chroma), we used binary units. Technically, this is not possible because the chroma features are continuous values that lie between 0 and 1. However, we can interpret these values as probabilities and sample from them, yielding binary input data. In practice, we do not perform this sampling explicitly, but we use the *mean*

[4] Excluding known duplicates and tracks used for validation and testing for any of the tasks.

field approximation (see Section 5.2.2). Learning is much more stable for binary units than for Gaussian units, so being able to use binary units is a significant advantage.

We used single step constrastive divergence (CD-1) everywhere. A learning rate of 0.005 was used to train the RBMs with binary visible units; a learning rate of 0.0001 was used for the RBM with Gaussian visible units. We performed only a single run through the entire dataset; performing multiple epochs turned out to be unnecessary (and would require too much computation time).

5.2.1 Sparsity

We modified the hidden unit activations according to [7] to encourage them to be sparse. Convolutional RBMs are overcomplete models, so adding a sparsity penalty term ensures that the learnt feature representations are useful [14]. In addition, sparse activations are essential for max-pooling to work properly [5, 17].

We used a sparsity target of 0.05 for layers 1a and 1b, and a target of 0.1 for layer 3. A relative sparsity cost of 0.1 was used in all cases.

5.2.2 Mean Field Approximation

Where possible, we eliminated sampling steps by using the mean field approximation. This eliminates sampling noise and often positively affects convergence. We used this for the chroma inputs and in the contrastive divergence algorithm, except when updating the hidden states, as recommended in [10]. Interpreting continuous input values that are constrained to a finite interval as input probabilities to train an RBM is common practice [9].

6. EXPERIMENTS

We trained the network as a convolutional MLP for each of the classification tasks described in Section 4: first with random initialization of the weights, and then using the weights learnt by the DBN (supervised finetuning), yielding six experiments. We tried learning rates of 0.05, 0.005 and 0.0005 and trained for 30 epochs. To initialize the random weights, we sampled them from a Gaussian distribution with a mean and variance corresponding to those of the weights learnt by the DBN. This ensures that the results are comparable.

We also trained a naive Bayes classifier and a logistic regression classifier that operate on windows of features from the dataset, resulting in six more experiments. We chose a window size of 32 beats (8 bars), which is comparable to the timescale on which the convolutional network operates. For the logistic regression classifier, we tried learning rates of 0.005, 0.0005, $5 \cdot 10^{-5}$, $5 \cdot 10^{-6}$ and $5 \cdot 10^{-7}$ and also trained for 30 epochs. Both the chroma features and the timbre features were normalized to have a zero mean and a unit variance in this case.

For each of the twelve experiments, we determined the optimal parameters using the validation sets, and then computed the classification accuracies on the test sets using these parameters. The results can be found in Table 1.

7. RESULTS

The first thing to notice is that the key detection task seems to be fairly simple. The achieved accuracies are much higher than for the other tasks, and even the simplest technique performs quite well. Windowed logistic regression performs best. There are multiple possible explanations for this:

- the property we are trying to determine is quite 'low-level'. The key of a track is in a very close relationship with the chroma features and how they evolve through time. Relating the genre or the artist to these features is much more difficult;
- to construct the dataset for this task, we selected tracks with a high key confidence. This implies that the algorithm used to annotate key information in the Million Song Dataset could identify the key of these tracks with relative ease. It would make sense that the same is true for our models. Unfortunately, there is no way to verify this, except by constructing a manually labeled dataset.

For the other tasks, the convolutional network has a definite edge over the other approaches: the classification accuracies increase significantly.

The gains obtained with pretraining on the other hand seem to be much more modest; this is only advantageous for the artist recognition task, which is quite difficult because it is a 50-way classification problem. The utility of pretraining for this task could stem from the fact that the number of tracks per class available for training (80) is much lower compared to the other tasks (200). Indeed, it has been shown that gains from unsupervised pretraining are maximal when the amount of available labeled training data is limited [6]. This data scarcity is inherent to the task at hand - few artists have a discography with more than 100 tracks.

The optimal learning rate for key detection with the convolutional network differs depending on whether pretraining is used or not. This is because the training for this task without pretraining did not converge after 30 epochs using a learning rate of 0.005. This indicates that convergence is faster when pretraining is used. To investigate this, we also compared classification accuracies obtained after only 20 training epochs, which can be found in the bottom half of Table 1. We now observe that pretraining is beneficial for all tasks. This confirms that it improves convergence speed.

8. CONCLUSION AND FUTURE WORK

We have trained a convolutional network on beat-aligned timbre and chroma features obtained from music audio data

		genre recognition	artist recognition	key detection
	naive Bayes	10.02%	6.80%	73.74%
30 epochs	windowed logistic regression	25.90% ($5 \cdot 10^{-6}$)	32.13% ($5 \cdot 10^{-5}$)	**86.53%** ($5 \cdot 10^{-5}$)
	conv. MLP without pretraining	**29.52%** (0.005)	34.34% (0.05)	83.84% (0.05)
	conv. MLP with pretraining	29.12% (0.005)	**35.74%** (0.05)	83.84% (0.005)
20 epochs	conv. MLP without pretraining	24.90% (0.05)	33.94% (0.05)	83.84% (0.05)
	conv. MLP with pretraining	**27.31%** (0.005)	**35.54%** (0.05)	**84.51%** (0.005)

Table 1. Test accuracies and corresponding learning rates for each of the classification tasks, with and without pretraining.

to perform a number of classification tasks. The convolutional nature of the network allowed us to summarize these features over musically significant timescales, leading to an increase in accuracy. We used unsupervised pretraining with a very large dataset, which improved convergence speed and, for the artist recognition task, classification accuracy. It is clear that the ability to harness a large amount of unlabeled data is advantageous for tasks where the amount of available training data is limited.

In future work, we would like to refine a couple of aspects about the architecture of the network, such as the way the input features are modeled in the lower layers: other types of visible units might be more suitable. We will also investigate different ways to encourage the RBMs to learn interesting features, besides the sparsity penalty term that we used for these experiments.

9. REFERENCES

[1] Yoshua Bengio. Learning deep architectures for AI. Technical report, Dept. IRO, Université de Montreal, 2007.

[2] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral.

[3] T. Bertin-Mahieux, R. Weiss, and D. Ellis. Clustering beat-chroma patterns in a large music database. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, 2010.

[4] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011. (submitted).

[5] Y-Lan Boureau, Jean Ponce, and Yann Lecun. A theoretical analysis of feature pooling in visual recognition. In *27th International Conference on Machine Learning*, 2010.

[6] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. pages 153–160, April 2009.

[7] Hanlin Goh, Nicolas Thome, and Matthieu Cord. Biasing restricted boltzmann machines to manipulate latent selectivity and sparsity. In *Deep Learning and Unsupervised Feature Learning Workshop — NIPS*, 2010.

[8] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks networks. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, 2010.

[9] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:2002, 2000.

[10] Geoffrey E. Hinton. A practical guide to training restricted boltzmann machines. Technical report, University of Toronto, 2010.

[11] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, 2006.

[12] Alex Krizhevsky. Convolutional deep belief networks on cifar-10. Technical report, University of Toronto, 2010.

[13] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, New York, NY, USA, 2009. ACM.

[14] Honglak Lee, Peter Pham, Yan Largman, and Andrew Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1096–1104. 2009.

[15] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. Multimodal deep learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[16] M. Norouzi, M. Ranjbar, and G. Mori. Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2735 –2742, 2009.

[17] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of the 20th International Conference on Artificial Neural Networks (ICANN)*, 2010.