LAST CHANCE LIT

JOSEPH BOYD

Contents

1. Neumann, Beate, et al. "Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes." Nature 464.7289 (2010): 721-727. 5
2. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014). 5
3. He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. 6
4. Ljosa, Vebjorn, et al. "Comparison of methods for image-based profiling of cellular morphological responses to small-molecule treatment." Journal of biomolecular screening 18.10 (2013): 1321-1329. 6
5. Myers, Gene. "Why bioimage informatics matters." Nature methods 9.7 (2012): 659. 7
6. Ganin, Yaroslav, and Victor Lempitsky. "Unsupervised domain adaptation by backpropagation." International Conference on Machine Learning. 2015. 8
7. Zhang, Ji-Hu, Thomas DY Chung, and Kevin R. Oldenburg. "A simple statistical parameter for use in evaluation and validation of high throughput screening assays." Journal of biomolecular screening 4.2 (1999): 67-73. 9
8. Loo, Lit-Hsin, Lani F. Wu, and Steven J. Altschuler. "Image-based multivariate profiling of drug responses from single cells." Nature methods 4.5 (2007): 445-453. 9
9. Swinney, David C., and Jason Anthony. "How were new medicines discovered?." Nature reviews Drug discovery 10.7 (2011): 507-519. 11
10. Orlov, Nikita, et al. "WND-CHARM: Multi-purpose image classification using compound image transforms." Pattern recognition letters 29.11 (2008): 1684-1693. 12
11. Uhlmann, Virginie, Shantanu Singh, and Anne E. Carpenter. "CP-CHARM: segmentation-free image classification made accessible." BMC bioinformatics 17.1 (2016): 51. 13
12. Haney, Steven A., et al. "High-content screening moves to the front of the line." Drug discovery today 11.19 (2006): 889-894. 13
13. Shay, Jerry W., and Woodring E. Wright. "Hayflick, his limit, and cellular ageing." Nature reviews Molecular cell biology 1.1 (2000): 72-76. 13

14. Dürr, Oliver, and Beate Sick. "Single-cell phenotype classification using deep convolutional neural networks." Journal of biomolecular screening 21.9 (2016): 998-1003. 14
15. Kandaswamy, Chetak, et al. "High-content analysis of breast cancer using single-cell deep transfer learning." Journal of biomolecular screening 21.3 (2016): 252-259. 14
16. Perlman, Zachary E., et al. "Multidimensional drug profiling by automated microscopy." Science 306.5699 (2004): 1194-1198. 14
17. Singh, Shantanu, Anne E. Carpenter, and Auguste Genovesio. "Increasing the content of high-content screening: an overview." Journal of biomolecular screening 19.5 (2014): 640-650. 15
18. Adams, Cynthia L., et al. "[24]-Compound Classification Using Image-Based Cellular Phenotypes." Methods in enzymology 414 (2006): 440-468. 16
19. Jones, Thouis, Anne Carpenter, and Polina Golland. "Voronoi-based segmentation of cells on image manifolds." Computer Vision for Biomedical Image Applications (2005): 535-543. 18
20. Godinez, William J., et al. "A multi-scale convolutional neural network for phenotyping high-content cellular images." Bioinformatics (2017): btx069. 19
21. Van Valen, David A., et al. "Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments." PLoS computational biology 12.11 (2016): e1005177. 19
22. Sadanandan, Sajith Kecheril, Petter Ranefall, and Carolina Whlby. "Feature augmented deep neural networks for segmentation of cells." European Conference on Computer Vision. Springer International Publishing, 2016. 20
23. Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015. 20
24. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, Cham, 2015. 21
25. Ciresan, Dan, et al. "Deep neural networks segment neuronal membranes in electron microscopy images." Advances in neural information processing systems. 2012. 22
26. Slack, Michael D., et al. "Characterizing heterogeneous cellular responses to perturbations." Proceedings of the National Academy of Sciences 105.49 (2008): 19306-19311. 22
27. Boland, Michael V., and Robert F. Murphy. "A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of HeLa cells." Bioinformatics 17.12 (2001): 1213-1223. 23

28. Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005. 23
29. Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps." arXiv preprint arXiv:1312.6034 (2013). 24
30. Avondoglio, Dane, et al. "High throughput evaluation of gamma-H2AX." Radiation Oncology 4.1 (2009): 31. 24
31. Mining of Massive Datasets 24
31.1. Chapter 2 - MapReduce and the New Software Stack 24
31.2. Chapter 3 - Finding Similar Items 25
31.3. Chapter 5 - Link Analysis 26
32. Eulenberg, Philipp, et al. ""Reconstructing cell cycle and disease progression using deep learning." bioRxiv (2017): 081364. 26
33. Louppe, Gilles, et al. "Understanding variable importances in forests of randomized trees." Advances in neural information processing systems. 2013. 26
34. Li, Fei-Fei, Karpathy, Andrej, and Johnson, Justin. "CS231n – Convolutional Neural Networks for Visual Recognition" (2016) 27
34.1. Lecture 1 – Introduction 27
34.2. Lecture 2 – Image Classification Pipeline 27
34.3. Lecture 3 – Loss Functions and Optimisation 28
34.4. Lecture 4 – Backpropagation and Neural Networks 28
34.5. Lecture 5 – Training Neural Networks, Part 1 29
34.6. Lecture 6 – Training Neural Networks, Part 2 30
34.7. Lecture 7 – Convolutional Neural Networks 31
34.8. Lecture 8 – Spacial Localisation and Detection 32
34.9. Lecture 9 – Understanding and Visualising Convolutional Neural Networks 32
34.10. Lecture 10 – Recurrent Neural Networks 32
34.11. Lecture 11 – CNNs in Practice 33
34.12. Lecture 12 – Software Packages: `Caffe`/Torch/`Theano`/`TensorFlow` 34
34.13. Lecture 13 – Segmentation and Attention 34
34.14. Lecture 14 – Videos and Unsupervised Learning 34
35. Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013). 35
36. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., ... & Lempitsky, V. (2016). Domain-adversarial training of neural networks. Journal of Machine Learning Research, 17(59), 1-35. 38
37. Frosst, N., & Hinton, G. (2017). Distilling a Neural Network Into a Soft Decision Tree. arXiv preprint arXiv:1711.09784. 42
38. Young, Daniel W., et al. "Integrating high-content screening and ligand-target prediction to identify mechanism of action." Nature chemical biology 4.1 (2008): 59-68. 42

39. Horbach, S. P., & Halffman, W. (2017). The ghosts of HeLa: How cell line misidentification contaminates the scientific literature. PloS one, 12(10), e0186281. 43
40. Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." Nature 323.6088 (1986): 533-538. 43
41. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International Conference on Machine Learning. 2015. 44
42. Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014. 45
43. Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size. arXiv preprint arXiv:1602.07360. 46
44. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 46
45. Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. 2010. 47
46. Gordon, Geoff, and Ryan Tibshirani. "Gradient descent revisited." Optimization 10.725/36 (2012): 725. 48
47. Schönenberger, Felix, et al. "Discrimination of cell cycle phases in PCNA-immunolabeled cells." BMC bioinformatics 16.1 (2015): 180. 49
48. Otsu, Nobuyuki. "A threshold selection method from gray-level histograms." IEEE transactions on systems, man, and cybernetics 9.1 (1979): 62-66. 50
49. Rose, France, et al. "Compound Functional Prediction Using Multiple Unrelated Morphological Profiling Assays." SLAS TECHNOLOGY: Translating Life Sciences Innovation (2017): 2472630317740831. 50
50. Bubeck, Sébastien. "Convex optimization: Algorithms and complexity." Foundations and Trends in Machine Learning 8.3-4 (2015): 231-357. 51
50.1. Chapter 1 - Introduction 51
51. Elkan, Charles. "The foundations of cost-sensitive learning." International joint conference on artificial intelligence. Vol. 17. No. 1. Lawrence Erlbaum Associates Ltd, 2001. 52
52. Dal Pozzolo, Andrea, et al. "Calibrating probability with undersampling for unbalanced classification." Computational Intelligence, 2015 IEEE Symposium Series on. IEEE, 2015. 53
53. Bengio, Yoshua. "The consciousness prior." arXiv preprint arXiv:1709.08568 (2017). 53

1. Neumann, Beate, et al. "Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes." Nature 464.7289 (2010): 721-727.

This study is motivated by the need to understand the connections between the 21000 genes in the human genome and some of the basic cellular functions, in pariticular, mitosis. For this purpose, a highly developed screening platform was used to analyse the phenotypic profiles of cell populations subject to *RNA interference*. This interference is achieved with small interference RNA (siRNA), synthetic RNA molecules that reduce gene expression mRNA by a significant amount (by an average of 87% in the study).

The data consisted of two days worth of time-lapse fluorescence microscopy, with the green fluorescent protein (GFP) tagging the core histone 2B, a protein within chromosomal histones. Hence, cell nuclei alone were recorded. The usual pipeline was used: segmentation, feature extraction (200 features), and classification of cell nuclei into 16 phenotype classes by an SVM trained on 3000 annotated samples and achieving 87% accuracy. As time-lapse microscopy, (not an "end-point" assay), mitotic states are more easily detected. Figure 1B shows interestingly how dominant the interphase class is (82.2%). For each phenotypic class, the proportion was tracked over time and compared with a negative control. The maximum difference in proportions over time was used as an index. Potential hits were identified to be siRNA yielding indices in at least four phenotypic classes below a certain threshold. Apart from the identification of hits, several analyses were conducted: event order maps were used to track the progression of phenotypes to discover the chronology and causality of each. Hierarchical clustering was performed on phenotypic profiles.

The study produced a publicly available dataset known as the MitoCheck dataset[1].

2. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

The Visual Geometry Group (VGG) at Oxford produced a streamlined CNN achieving greater depths and taking second place in the ImageNet 2014 competition. They focussed on a simple 3×3 kernel in every layer, with blocks of convolutional layers separated with max pooling, and the architecture ending with affine layers. In effect, it is just like a deeper AlexNet, but with 3×3 convolutional layers used everywhere. As they note, though the effective receptive field of a single 7×7 layers and three 3×3 layers are the same, the former has $49C^2$ parameters for C activation maps, whereas the latter has only $27C^2$. As they note,

> [VGG Net[2] architecture] did not depart from the classical ConvNet architecture of LeCun" [unlike GoogLeNet with its "inception" layers] (page 8)

and, as such, can be see as a bridge between AlexNet and ResNet.

[1] http://www.mitocheck.org/

[2] A name attributed afterwards

3. He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

ResNet represents an innovation on classical CNNs to address the "degradation" problem of training very deep networks. This phenomenon is distinct from overfitting as it refers to a degradation of *training error* when more and more layers are added. VGG showed that depth is usually more important than width (larger kernels), and ResNet is a successful attempt to reformulate CNNs to achieve extreme depths. The authors hypothesise that the way to achieve superior generalisation in deep neural nets is to make small changes to the feature representation over many layers rather than a dramatic transformation of inputs: "multiple nonlinear layers can asymptotically approximate complicated functions" (page 3). Therefore, they introduce residual learning where, after each two layers[3] of convolutions, the outputs are computed as,

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x},$$

where $\mathcal{F} = W_2\sigma(W_1\mathbf{x})$. They observe that it is much easier for a network to find $||\mathbf{F}||_2 \approx \mathbf{0}$ (i.e. the residual) than to find $\mathcal{F}$ that creates $\hat{\mathbf{x}} \approx \mathbf{x}$, that is, an effective identity mapping, which is what the network needs for good generalisation. Despite the benefits, adding the input comes at negligible cost to compute, and does not affect backpropagation, as it is a constant for future layers.

They present results on the ImageNet 2015 challenge where they won first prize, and also results on their theory that ResNets make smaller contributions per layer. They also run ResNet on CIFAR-10 with up to 1202 (!) layers. They finally discuss ResNet's use as the CNN in object detection frameworks (*Faster R-CNN*).

4. Ljosa, Vebjorn, et al. "Comparison of methods for image-based profiling of cellular morphological responses to small-molecule treatment." Journal of biomolecular screening 18.10 (2013): 1321-1329.

This is a survey of different approaches to image-based profiling, that is, characterising cell populations by some index as a means of predicting the mechanism of action[4] (MOA) of a particular compound. As they state,

> Image-based screens for particular cellular phenotypes are a proven technology contributing to the emergence of high-content screening as an effective drug- and target-discovery strategy [...] may help reduce the high levels of late-stage project attrition associated with target-directed drug-discovery strategies (page 2)

[3]Larger groups are also tried, but if it were every layer, it would just be a linear mapping (prior to the activation function)

[4]Mechanism of action is a pharmacological term referring to the biochemical reaction between the drug and the organism. The biochemical reaction usually makes reference to an enzyme or receptor protein.

> Profiling cell-based phenotypes is the next challenge for quantitative microscopy. The principle of phenotypic profiling is to summarize multiparametric, feature-based analysis of cellular phenotypes of each sample so that similarities between profiles reflect similarities between samples (page 2)
>
> Profiling is well established for biological readouts such as transcript expression and proteomics. Comparatively, image-based profiling comes at a much lower cost, can be scaled to medium and high throughput with relative ease, and provides single- cell resolution. (page 2)

Thus, the authors extract the "core profiling methods" from five methodologies that create "per-sample" profiles from "per-cell" measurements. They test them on the MCF-7 cell line on 96-well plates, treated with 113 compounds at 8 concentrations in triplicate ($\implies$ 29 plates at least) and marked with fluorescent tags. The resulting images were analysed with CellProfiler[5] producing 453 features for each cell. The methods were used to create profiles of a certain size, and the element-wise median over the triplicate was used. The prediction of mechanism of action was by nearest neighbour (1-NN) in cosine distance,

$$d(p, q) = 1 - \cos \theta_{p,q}$$

from a pool of labelled compounds. The five methods are each quite distinct. The most interesting is perhaps the "normal to SVM hyperplanes". Here the profile is the normal vector of the SVM hyperplane separating the treated sample cells and a negative control (DMSO). Dimensionality reduction was done by training the SVM on all D measurements, then $D-1, D-2, \ldots, 1$, with the dimensionality chosen to be the one maximising accuracy. This scheme is known as SVM recursive feature elimination (SVMRFE). Despite this and other sophisticated approaches, the best approach was simply to take the mean of each of the 453 features over the whole population, with a predictive accuracy of 88%.

5. Myers, Gene. "Why bioimage informatics matters." Nature methods 9.7 (2012): 659.

In this short article from 2012, prominent researcher Gene Myers characterises bioimage informatics as an emerging field of analysis offering insights into cell structure and behaviour lost in classical -omics data, giving examples thereof. He paints the bioimage informatician as either a computer vision practitioner seeking new problems, or biologists embracing the world of data science. He contends bioimage informatics matters for two reasons: because of the production of large image data sets and because of the trend in bioinformatics to answer questions pertaining to spatial properties of cells.

> From my perspective, it is very reminiscent of the state of bioinformatics in the early 1980s: the exciting, somewhat chaotic free-for-all that is potentially the birth of something new. (page 1)

[5] http://cellprofiler.org/

6. Ganin, Yaroslav, and Victor Lempitsky. "Unsupervised domain adaptation by backpropagation." International Conference on Machine Learning. 2015.

This paper develops a new approach to domain adaptation in feed-forward neural nets, where a large labelled source set is available and an unlabelled (or partly labelled) target set is available[6]. The approach is compatible with any feed forward network. It involves training a classifier for the source data set, and a classifier to distinguish between domains, as part of the same loss function. The parameter set, $\theta = \{\theta_f, \theta_y, \theta_d\}$ consists of the weights for the shared deep feature layers, θ_f, the weights of the classifier layers, θ_y, and the weights of the domain discriminator, θ_d. The domain discriminator loss is scaled by a negative tuning parameter, λ. Hence, minimising the full loss function for the deep features θ_f *maximises* the loss of the discriminator. Thus, deep features are chosen that are invariant between the domains, as they are chosen to be maximally indistinguishable between the domains. Thus, features that are both useful for classification and invariant to the domain shift are discovered. Simultaneously, the parameters θ_d are chosen to minimise the loss. Clearly, if they also maximised the loss, there would be no onus on the deep features to be invariant. Formally, the loss function is $\mathbb{E}(\theta_f, \theta_y, \theta_d) = \mathcal{L}_y(\theta_f, \theta_y) - \lambda \mathcal{L}_d(\theta_f, \theta_d)$. Then,

$$
\begin{aligned}
\hat{\theta}_f &= \min_{\theta_f} \mathbb{E}(\theta_f, \hat{\theta}_y, \hat{\theta}_d) \\
&= \min_{\theta_f} \mathcal{L}_y + \lambda \min_{\theta_f} \{-\mathcal{L}_y\} = \min_{\theta_f} \mathcal{L}_y + \lambda \max_{\theta_f} \{\mathcal{L}_y\} \\
\hat{\theta}_y &= \min_{\theta_y} \mathbb{E}(\hat{\theta}_f, \theta_y, \hat{\theta}_d) = \min_{\theta_y} \mathcal{L}_y \\
\hat{\theta}_d &= \max_{\theta_d} \mathbb{E}(\hat{\theta}_f, \hat{\theta}_y, \theta_d) = \max_{\theta_d} \{-\mathcal{L}_d\} = \min_{\theta_d} \mathcal{L}_d
\end{aligned}
$$

These lead to the set of rules for optimisation,

$$
\theta_f \leftarrow \theta_f - \mu \left(\frac{\partial \mathcal{L}_y^i}{\partial \theta_f} - \lambda \frac{\partial \mathcal{L}_d^i}{\partial \theta_f} \right)
$$

$$
\theta_y \leftarrow \theta_f - \mu \frac{\partial \mathcal{L}_y^i}{\partial \theta_y}
$$

$$
\theta_d \leftarrow \theta_f - \mu \frac{\partial \mathcal{L}_d^i}{\partial \theta_d}
$$

[6]Note that the difference between domain adaptation and transductive transfer seems very slight: in this study (domain adaptation), features are learned with respect to both domains so that they will be useful to both. Nevertheless, in the end, a single classifier is learned. In an example of transductive transfer, a transfer function is learned over the parameters of source classifiers, and used to create target classifiers directly, based only on the (unlabelled) target domain. The fact that the source and target models are distinct seems to be the only difference.

However, in practice SGD solvers are cannot both minimise and maximise a loss function. Therefore, the authors introduce a *gradient reversal layer*, defined by,

$$R_\lambda(\mathbf{x}) = \mathbf{x}$$

$$\frac{\partial R_\lambda}{\partial \mathbf{x}} = -\lambda \mathbf{I}$$

noting that the derivative is not "compatible", to reformulate the loss function such that the gradients are correctly calculated by any standard SGD solver as a matter of course. This amounts to a computational hack. While it is indeed true that a deep learning framework such as `TensorFlow` allows the user to specify the gradient arbitrarily, it is not clear why the user could not just specify the update rules manually instead. The paper concludes with the presentation of results and t-SNE visualisation of how the features distributions overlap after domain adaptation.

7. Zhang, Ji-Hu, Thomas DY Chung, and Kevin R. Oldenburg. "A simple statistical parameter for use in evaluation and validation of high throughput screening assays." Journal of biomolecular screening 4.2 (1999): 67-73.

Any HTS assay is subject to random variation as well as systematic measurement error. A typical validation step in HTS is to assure statistics measured between controls and test samples can be distinguished with confidence. The *Z-factor* expresses the ratio of the separation band $|\mu_c - \mu_s| + 3\sigma_c + 3\sigma_s$ (that is, the distance between the opposing tails of the distributions at three standard deviations) and the dynamic range, $|\mu_c - \mu_s|$ between the control c and the test sample s. Thus, the Z factor,

$$Z = 1 - \frac{3\sigma_c + 3\sigma_s}{|\mu_c - \mu_s|}$$

The maximum (ideal) value is therefore 1. In the field, it is an accepted fact that anything below 0 is considered as a readout lacking robustness. The related Z'-factor calculates the same statistic for the positive and negative controls. The positive and negative controls should represent the high and low of the range of measurements of a property of interest. The Z'-factor thus refers to the overall quality of the assay.

8. Loo, Lit-Hsin, Lani F. Wu, and Steven J. Altschuler. "Image-based multivariate profiling of drug responses from single cells." Nature methods 4.5 (2007): 445-453.

In this monumental paper, the authors introduce a thorough framework for phenotypic profiling. The framework centers around using a linear SVM to compare test compounds at (13) different dosage levels with a negative control (DMSO) in a large assay. The

normal[7] to the separating hyperplane (the SVM weight vector) is used to characterise the test compound-dosage combination. 100 compounds are tested at 13 different levels, giving a *titration*[8] series for each. The negative control is dimethyl sulfoxide (DMSO) alone (noting that all tests use a DMSO substrate). The cells are segmented using the watershed transformation[9] and several hundred features are extracted per cell. These include Haralick and Zerneke features[10]. The dimensionality is reduced using the technique SVMRFE (recursive feature elimination), which is a backward elimination wrapper method for feature selection (Guyon et alii). The profiles are clustered (titration clustering) with multiple clusters indicating *multiphasic* responses at different dosage levels. The cluster constituents are aggregated to form a representative dosage range profile or *d-profile*. The profiles are then used for:

> Typical applications of high-throughput image- based assays, such as drug screening, phenotypic change detection and category prediction(page 3)

The study encompasses a rich assortment of analyses and heuristic decisions that are too numerous to list. Two interesting techniques are multidimensional scaling (MDS)[11] for visually comparing the profile clusters and kernel density estimation[12] (KDE) in the resampling procedure. Note that this study predates the development of CellProfiler, and much of the software was custom made by the authors.

---

[7]A reminder on normal vectors: the normal vector of a function $z = f(x, y)$ is $\mathbf{n} = [f_x, f_y, -1]^T$. This is equivalently the gradient of an auxiliary function, $g(x, y, z)$ with *level curve* defined by the relation $f(x, y) - z = 0$. Recall the gradient, ∇f of a function is always normal to the local level curve or contour of that function. This may be seen from the definition of *directional derivative*, defined as $\nabla_{\mathbf{v}} f(\mathbf{x}) = \lim_{h\to 0}(f(\mathbf{x} + h\mathbf{v}) - f(\mathbf{x}))/h$. This quantity is 0 for a choice of $\mathbf{v}$ to be tangential to the level curve, since then $f(\mathbf{x} + h\mathbf{v}) = f(\mathbf{x})$. Furthermore, $\nabla_{\mathbf{v}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{v}$, and therefore the gradient is normal to the tangent vector of the level curve. N.B. this is most easily observed visually.

[8]A titration is an experimental process for determining the concentration of a certain substance by incrementally adding a reactant.

[9]The watershed transformation is a technique for image segmentation. It works by "flooding" the topological landscape of the (inverted) image. Where catchment basins intersect, one marks *watershed lines* that correspond to the segmentation contours.

[10]Haralick features derive from the co-occurrence matrix of an image. The co-occurrence matrix is a matrix whose elements count the number of times pairs of pixel intensities occur in the image at a given offset. The elements are thus indexed by the pixel values themselves, and hence the co-occurrence matrix is square with dimension the dynamic range of the image. Haralick features somehow derive from this and are used to measure texture. Zernike features are features based on the Zernicke polynomials that describe shape.

[11]Multidimensional scaling is a technique for visualising the similarities of data points in lower dimensions. It is thus similar to t-SNE or Sammon mapping.

[12]Kernel density estimation or *Parzen window* density estimation is a non-parametric technique for inferring the probability density function generated a set of sample observations. It is done by choosing a kernel (probability distribution) that is placed at each observation. The estimate is the sum of all these overlapping functions. It is thus similar to a histogram. The choice of kernel and bandwidth (variance) parameter is made by the implementer.

9. Swinney, David C., and Jason Anthony. "How were new medicines discovered?." Nature reviews Drug discovery 10.7 (2011): 507-519.

The authors indicate that despite an increase in funding in drug screening, the annual number of FDA-approved[13] innovative drugs (termed "first-in-class") does not rise accordingly. They attribute this to the predominance of target-based approaches in drug discovery:

> Since the dawn of the genomics era in the 1990s, the main focus of drug discovery has been on drug targets (page 1)

Target-based approaches encompass a principled approach to drug discovery: identify a target protein that plays a role in a disease, and design a treatment to alter this protein. RNA interference is one tool involved in the identification of targets, whereby targeted mRNA is down regulated and the effects (transcriptomic, phenotypic, etc.) noted. On the other hand, these hypotheses may be based on spurious or incomplete evidence: often there is not just one smoking gun gene behind disease pathogenesis[14], rather a combination of genes. The authors indicate that the marginalisation of phenotypic screens by the pharmaceutical industry has resulted in the stagnation of innovation drug treatments, with a high *attrition rate* in the process.

> A strength of the phenotypic approach is that the assays do not require prior understanding of the molecular mechanism of action (MMOA), and activity in such assays might be translated into therapeutic impact in a given disease state more effectively than in target-based assays, which are often more artificial. A disadvantage of phenotypic screening approaches is the challenge of optimizing the molecular properties of candidate drugs without the design parameters provided by prior knowledge of the MMOA. An additional challenge is to effectively incorporate new screening technologies into phenotypic screening approaches, which is important for addressing the traditional limitation of some of these assays: a considerably lower throughput than target-based assays. (page 2)

In summary, it is perhaps more effective to try out many likely compounds and see what happens than to try to identify a particular one. Where target-based approaches are more effective, however, is in the creation of *follower drugs*, third party equivalents to first-in-class drugs. Note the *molecular mechanism of action* (MMOA) refers to the specific molecular interaction between the drug and the target. Mechanism of action (MOA) refers to a physiological response (e.g. anti-inflammatory).

The study looks at 257 drugs published between 1999 and 2008. The findings are that, despite the preeminence of target-based approaches, the most common mode of first-in-class drug discovery is phenotypic screening. This is most true of infectious and central nervous

[13]FDA is the Food and Drug Administration, a wing of the US government responsible for regulating food safety, and drug sales.

[14]Either the biological mechanisms responsible for or the development of a disease.

system diseases. Cancer treatments are most frequently discovered by biologics[15], which also predominate for diseases of the immune system. Target-based approaches succeed for the discovery of half of the follower drugs. The authors postulate the high attrition rate of target-based approaches to be due to the failing of one or more of three hypotheses that must all succeed for a discovery to be made:

> The first hypothesis, which also applies to other discovery approaches, is that activity in the preclinical screens that are used to select a drug candidate will translate effectively into clinically meaningful activity in patients. The other two hypotheses are that the target that is selected is important in human disease and that the MMOA of drug candidates at the target in question is one that is capable of achieving the desired biological response. (page 9)

The two main weaknesses of phenotypic screens are identified as being: the necessity of finding the MMOA *a posteriori* and; the lower throughput bottleneck.

This paper also uses a lot of useful terminology: preclinical strategies, potential drug candidates, target-based, phenotypic screens, modification of natural substances, biologic-based approaches, molecular mechanism of action (MMOA), pharmacological response, allosteric or orthosteric.

10. Orlov, Nikita, et al. "WND-CHARM: Multi-purpose image classification using compound image transforms." Pattern recognition letters 29.11 (2008): 1684-1693.

Weighted neighbor distances using a compound hierarchy of algorithms representing morphology (WND-CHARM) is a classification framework designed to be versatile over many problems. It works by extracting a large number of image features (> 1000) before weighting them using Fischer scores (soft thresholding). The features with lowest weights are then discarded (hard thresholding). Test samples are then classified with a 1-NN with a particular distance measure. This they call weighted neighbour distances (WND).

> The advent of high content screening (HCS) where the goal is to search through tens of thousands of images for a specific target morphology requires a flexible classification tool that allows any morphology to be used as a target.

(page 1)

WND-CHARM is compared to tailored models on various datasets, including the **HeLa dataset** where it performs favourably.

[15]Biologics are genetically-engineered proteins that target the immune system.

11. Uhlmann, Virginie, Shantanu Singh, and Anne E. Carpenter. "CP-CHARM: segmentation-free image classification made accessible." BMC bioinformatics 17.1 (2016): 51.

> Segmentation-free image classification using whole-image features has already been widely used in computer vision, especially for image databases [1, 2]. It is however less popular for bioimage analysis, where segmentation remains the most common paradigm.

This is a confusing point to make. Segmentation and whole/full image classification were never a focus of WND-CHARM. Even for the biological datasets, they were all pre-segmented anyway (the HeLA dataset consists of crops of cells). Therefore, CP-CHARM is also reinterpreting the purpose of WND-CHARM. CP-CHARM is an adaptation of WND-CHARM to comprise more conventional technologies. It replaces the custom feature extraction with CellProfiler (this is what CP stands for), feature weighting is replaced with PCA, WND is replaced with LDA, and 75 : 25 cross validation is replaced with 10-fold cross validation.

12. Haney, Steven A., et al. "High-content screening moves to the front of the line." Drug discovery today 11.19 (2006): 889-894.

In this older paper, the authors describe how, with the advancement of relevant technologies, high content screening (HCS) has moved to the early stages of preclinical screening[16], and *hit-to-lead* stages. The authors identify the strengths of HCS are in its precision: multivariate per-cell analysis. Assays are further *multiplexed* with multiple fluorescent probes (usually up to 3 or 4).

13. Shay, Jerry W., and Woodring E. Wright. "Hayflick, his limit, and cellular ageing." Nature reviews Molecular cell biology 1.1 (2000): 72-76.

This short article traces the history of the *Hayflick limit*, the upper limit on the number of possible cell divisions. This is due to the eventual loss of the *telomeres* at the ends of the DNA strands which finally causes senescence, a cessation of mitosis. This originates with American medical scientist Leonard Hayflick (1928-), who defied the earlier assertion of eminent scientist Alexis Carrel:

> The largest fact to have come from tissue culture in the last fifty years is that cells inherently capable of multi- plying will do so indefinitely if supplied with the right milieu in vitro. (page 2)

Cancel cells therefore need to maintain telomeres to continue metastasis.

[16]Preclinical refers to experiments done before *clinical trials*, that is, tests involving humans.

14. Dürr, Oliver, and Beate Sick. "Single-cell phenotype classification using deep convolutional neural networks." Journal of biomolecular screening 21.9 (2016): 998-1003.

In this simple study, the authors pit a CNN against shallow models trained on CellProfiler-extracted features for classification of 40,000 images from the "cell painting" assay, BBBC022v1. This is the U2OS cell line–human bone epithelial cells. The classification task is to identify the MOA (MMOA?) of a drug from four classes (including the null–DMSO) given a cropped cell nucleus. **Note this is different from classifying the cell phenotype itself.** The dataset is imbalanced, with a preponderance of DMSO training samples. The CNN outperforms (by a dubious amount) the best of the three shallow models–LDA. The CNN architecture resembles VGG net, albeit with an input size of $(5 \times 72 \times 72)$ (a channel for each of the five stains). It would probably play more to the strengths of a CNN to classify phenotype itself.

15. Kandaswamy, Chetak, et al. "High-content analysis of breast cancer using single-cell deep transfer learning." Journal of biomolecular screening 21.3 (2016): 252-259.

Refers to high content analysis assays and high-throughput image-based assays interchangeably. Here, the BBBC021 dataset is used, the same as in Ljosa et al. (2013). They divide the 12 MOA class dataset into two (6 MOA classes each). They train stacked autoencoders (which they call autoassociators for some reason) on the 453-dimensional data (extracted with CellProfiler N.B. this is not pixel data). This unsupervised learning learns a succession of hidden layers, each compressing the last as an autoencoder. The learned weights are then used to initialise a deep architecture with a (6-way) logistic output that is fine-tuned on the labeled data. This alone has good performance on per-cell MOA classification. They try transferring layers from each domain to the other ($P_1 \to P_2$ and $P_2 \to P_1$) at four different depths. The results are varied and generally unconvincing. In fact, this contrived transfer framework feels rather wanton. Their comparison with an SVM (20% accuracy??) is bizarre. They further contrast their work with the profiling techniques for MOA prediction, but without clarifying how their method makes a population-level prediction of MOA. Presumably by using the mode prediction.

16. Perlman, Zachary E., et al. "Multidimensional drug profiling by automated microscopy." Science 306.5699 (2004): 1194-1198.

This is a short yet important paper. It studies 100 compounds with different known toxicity or therapeutic mechanisms in cancer, at different dosage levels. It is seminal foremostly for the following reason:

> To date [2004], drug effects have been broadly profiled with transcript analysis, proteomics, and measurement of cell line dependence of toxicity [...] Here, we suggest that large sets of unbiased measurements might serve as

> high-dimensional cytological profiles analogous to transcriptional profiles. We present a method that is hypothesis-free [...] (page 2)

although they only use 93 features. They are an obvious inspiration for, among others, the Loo paper.

> [...] profiling should be performed as a function of drug concentration.(page 2)

Their profile consist of a K-S statistic comparing the test cells to the control cells, for each feature. This gives a vector of K-S statistics that they further normalise to give their titration-invariant similarity score (TISS) (though this is not derived in the body).

They interestingly note that a histogram of total DNA content ought to be bimodal, given the interplay of cell cycle phases. Furthermore,

> G_2 and M populations may be distinguished by 2D display of total DNA signal against nuclear area. (page 2)

It also helpfully uses interesting technical terms: perturbations, systems level, micromolar to picomolar, "cultured in 384-well plates to near confluence, treated with drugs for 20 hours, fixed, and stained with fluorescent probes for various cell components and processes.", multiplexing, control population, arrested state, specificity and affinity, substrate.

17. Singh, Shantanu, Anne E. Carpenter, and Auguste Genovesio. "Increasing the content of high-content screening: an overview." Journal of biomolecular screening 19.5 (2014): 640-650.

This review remarks on the fact that although HCS articles become steadily more numerous from 2000 to 2012, the number of features used in the studies does not appear to increase, or at least far more slowly. Thus:

> [...] there is a strong tendency for HCS assays to typically be, in truth, quite low content. (page 2)

To demonstrate this, 118 papers were sampled fairly according to three search term strategies and their contents scrutinised. When visualised, their is only a weak trend towards higher content in HCS studies, despite a major rise in the amount of HCS research done. One interesting explanation for the low content in HCS studies is the adoption of the Z'-factor as an assay quality measure, whereas this was originally intended as an HTS property. There ensues an interesting discussion about why the Z'-factor may be unsuitable for HCS. It is is problematic because measurements are collected at the cell level, and are then aggregated over phenotypic sub-populations. The lower throughput HCS assays may further mean controls are not replicated enough to meaningfully compare the distributions of their readouts. The authors admonish the use of the Z'-factor in the conclusion, though without concluding on a suitable replacement. The latter section summarise the five profiling methods compared in Ljosa et al. (2013).

> We predict that advanced data analysis methods that enable full multiparametric data to be harvested for entire cell populations will enable HCS to finally reach its potential. (page 2)

18. Adams, Cynthia L., et al. "[24]-Compound Classification Using Image-Based Cellular Phenotypes." Methods in enzymology 414 (2006): 440-468.

The very first sentence of the abstract is the fundamental assumption on which all else is predicated:

> Compounds with similar target specificities and modes of inhibition cause similar cellular phenotypes. (page 1)

The aim of the study is to classify compounds into 12 classes of mechanism of action, which they do to an accuracy of about 90% (45/51). This is motivated by the benefits of detecting unintended compounds in the early, *in vitro* stages of screening, prior to clinical trials. This has the added ethical motivation of reducing the number of trials with unknown effects carried out on animals. They again make the case for cytoskeletal biology and morphological-based screens, pointing out the phenomena that evade observation at the genetic and proteomic levels.

For this purpose, the authors, who were part of a firm called Cytokinetics, developed a system called Cytometrix Technologies that may or my not still exist. The Cytometrix system comprises all the stages of high content screening: seeding of cells, application of fluorescent markers, microscopy, and image analysis software. The aim is to compute a phenotypic signature for each compound across different incubation periods, drug concentrations, and genotypes (cell lines).

> Changes in cell cycle, cell shape, cytoskeleton organisation, and protein trafficking and machinery in response to compounds often results in concomitant cellular morphological changes. (page 2)

They argue that a smaller number of fluorescent markers over a greater number of cell lines will give a more robust characterisation, unlike in Perlman et al. (2004), which measured many markers on a single cell line. The following subsections detail the protocol: the cell lines used; the cell plating–24h incubation at body temperature; preparation of compounds–triplicates of each compound at 8 concentrations including positive controls, and except negative control DMSO (single concentration); compound addition; immunocytochemistry–pipetting, fixing (formaldehyde), staining, washing; and microscopy.

They go on to describe the features measured, which include various shape-, texture-, and intensity-related measurements. These features are averaged by the strata of cell cycle phase: interphase, mitotic, preanaphase, as well as the proportions of each cell cycle phase. In this way, approach encompasses our own, though the classification of cycle phase is not based on supervised learning; rather on uni- or bivariate classification.

These population- and sub-population-level attributes are normalised (the features are of course on different scales) as standard deviations to the negative control attributes. Various comparisons can be made. One is in comparing attributes of each cell line w.r.t. each treatment (Figure 3). Another mode of comparison is in dose-response curves (titration series).

To classify compounds, they take the normalised attributes for each cell line and concatenate them as a row (32 attributes $\times$ 6 cell lines). Rows corresponding to different compound-concentration combinations form a matrix. PCA is performed on this matrix to reduce the compound-concentration signatures to 2-3 dimensions. These are visualised in Figure 5.

A concentration-independent measure of comparing drugs is created as the integral of the angle between dose responses of two compounds over a range of common dosage levels. The integral is necessarily discrete as we only have a small number of discrete dosage level measurements. Their strategy is then to use the signature data (reduced dimensionality) to cluster (Figure 6) and classify the data. Classification involves firstly identifying an subset of attributes, and then to predict the MOA using a nearest-centroids classifier based square distance between signatures. This they do to a high degree of accuracy (88%).

Their conclusions are varied, the main findings being that 1) measuring fewer markers over multiple cell lines is a viable approach to quantifying phenotypic differences 2) phenotypes can change significantly before cell mortality kicks in.

At this point, it really seems the Adams approach is not well represented in the Ljosa survey (although they do note that), even though it performs well. There, the 459 CellProfiler features are simply averaged and used as a profile directly. Adams takes a well-defined set of per-cell features, uses a primitive classifier identify cell cycle sub-populations and stratify, then normalise w.r.t. negative controls, then identify an optimal subset with LOOCV on a nearest-centroids classifier. If we run with the simple approach of averaging, consider the following argument:

We denote our phenotypic profile as $\mathbf{p} \in \mathbb{R}^K$ for K phenotype classes. Observe that the manually defined phenotype ontology acts as a latent variable behind the distribution of features. When it comes to the straightforward approach of phenotype characterisation in Ljosa et al. (2013) (inspired by Adams et al. (2004)), the phenotype profile $\mathbf{p}' \in \mathbb{R}^D$ for D features is given as $\mathbf{p}' = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$ for the N observations in the data set. Yet, under the hypothesis of a latent phenotype variable,

$$\mathbf{p}' \to \sum_{k=1}^{K} p_k \mathbf{f}_k$$

as $N \to \infty$, where the p_k are the proportions of each of the K phenotypes as measured by our approach, and the $\mathbf{f}_k$ are the average feature vector for each phenotype–the archetypal phenotypes, as it were. As such, $\mathbf{p}' \to \mathbf{F}\mathbf{p}$ as $N \to \infty$, with $\mathbf{F} \in \mathbb{R}^{D\times K}$ the matrix of $\mathbf{f}_k$. Thus, the Adams profile is approximately equivalent to our own projected into a high dimensional linear space. It therefore seems unlikely that cosine distances calculated by the former approach will be as meaningful as with our own.

The difference between our profiling approach and the surveyed once is that our require manual intervention (annotation). The other techniques are indices taken from already available information. Even the approach of Loo et al. (2007) that relies on an SVM, it is a binary classifier trained to distinguish sample populations from negative controls, and

the training data is by definition already annotated. A learning approach that circumvents this problem would need to be based on unsupervised or transfer learning techniques.

19. Jones, Thouis, Anne Carpenter, and Polina Golland. "Voronoi-based segmentation of cells on image manifolds." Computer Vision for Biomedical Image Applications (2005): 535-543.

This paper presents an approach in the domain of image cytometry[17] for segmenting cell borders based on previously identified cell nuclei "seeds". The approach contrasts with other approaches based either on a fixed offset around nuclei (imprecise), or on a watershed (unstable). The approach consists of defining a specialised distance metric that accounts for changes in intensity (high gradients), with the matrix,

$$\mathbf{G} = \begin{bmatrix} (\frac{\partial g}{\partial x})^2 & \frac{\partial g}{\partial y} \cdot \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial x} \cdot \frac{\partial g}{\partial y} & (\frac{\partial g}{\partial y})^2 \end{bmatrix}$$

where $\mathbf{g}(\mathcal{I})$ is a Gaussian blur[18] that stabilises the image measurements. The image gradients are simply the finite central difference of the adjacent pixels (these too can be defined by convolutions). This Riemannian metric[19] is combined with a regularisation matrix. It is easily demonstrated that the combined metric is simply a Euclidean distance in the limit. Hence, the segmentation becomes a Voronoi tesselation in the limit. The distance metric is defined for neighbouring pixels. Distances between non-adjacent pixels are computed as the shortest path (for example with Dijkstra's algorithm). Thus, the metric behaves like a spatial distance metric on flat, uniform regions, yet takes intensity into account when it matters.

The tunable components/parameters are therefore:

- The size (receptive field) of the Gaussian blur function, $g\mathcal{I}$. The authors use radii of 3 and 5.

[17] The image-based measurement of cell characteristics.

[18] An image convolution where the kernel distributes the weights according to a two-dimensional Gaussian.

[19] Some reminders: a metric is a distance function defined on a set of points. It satisfies the four properties: non-negativity, identify, symmetry, and the triangle inequality. A metric and set of points constitute a *metric space*, and the metric induces a certain *topology* on the set of points. A *manifold* is a space that is locally homeomorphic to Euclidean space, for example a sphere. Recall a geodesic is the equivalent of a straight line on curved spaces, for example, the "great circles" on the surface of a sphere. A Reimannian manifold is a manifold endowed with additional properties. Topological spaces generalise metric spaces and manifolds. Geometries are systems founded on axioms of shapes. Euclidean geometry was the original geometry, characterised by five properties. Euclidean geometry is part of a broader system articulated in the *Elements*, which is foundational to all of mathematics. The Cartesian plane, which assigns a pair of coordinates to every point in the two-dimensional Euclidean plane, unified geometry and algebra, revolutionising mathematics in the 17th century. Non-Euclidean geometries occur when the fifth postulate on parallel lines is relaxed, producing either hyperbolic or elliptical geometry. The former allows infinitely many non-intersecting non-parallel lines, and the latter allows none. Hyperbolic geometry, for example, is the geometry of a saddle surface–a surface with at least one saddle point (a stationary point that is, for example a local minima in one axis and local maxima in another).

- An initial global thresholding step to eliminate the background. This can be done with an Otsu thershold.
- The regularisation λ. They set it to 0.05 times the distance [difference?] between the average foreground and background pixel intensities on a per-image basis.

They evaluate their technique on a set of 16 images, with 80 cells on average per image, and over 20000 pixels of cell-cell edges (the most difficult case). They show samples of the approach working well compared to an (expert) manual segmentation, as well as some failure cases. They show how the vast majority of segmented boundaries are with 2 pixels (either way) of the ground truth with a histogram and cumulative distribution.

There are some choice quotes in the work too:

> Identification of individual cells allows much more powerful analysis of the resulting data than methods that provide only mean measurements for cell populations. For example, expression data from a protein interaction chip cannot differentiate a bi-modal population and a unimodal popuplation if they have the same mean expression levels. Measurement of individual cells prevents conflation of subpopulation. (page 1)

20. Godinez, William J., et al. "A multi-scale convolutional neural network for phenotyping high-content cellular images." Bioinformatics (2017): btx069.

This paper replaces the whole HCS pipeline (viz.[20] segmentation, feature extraction etc.) with a deep neural network termed "multi-scale" CNN (M-CNN), which processes whole images at 7 different resolutions, feeding them through the network in parallel convolutional pathways, which are finally concatenated for the final, affine layers. The authors apply the model in various datasets, in particular several of the BBBC datasets, for different classification problems. It is found to be highly successful, beating baseline CNNs AlexNet and GoogleNet, as well as traditional methods for per-image classification e.g. CP-CHARM. The most intriguing statement comes at the end:

> Typical deep learning architectures, such as AlexNet and GoogleNet, required a large amount of labeled data for training. Here we have showed that, together with a data augmentation strategy, relatively few images are required to train the M-CNN architecture. (page 8)

21. Van Valen, David A., et al. "Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments." PLoS computational biology 12.11 (2016): e1005177.

This paper presents an approach to segmentation of cell membranes using CNNs for per-pixel classification. The approach is less dubious than U-Net. The approach involves taking small crops around each pixel, labelling the crop as either `interior`, `border`, or `exterior`, and training a neural net. A relatively small number of crops may be augmented (on the fly)

[20] "videlicet", meaning namely

with rotation and reflection, creating a large training set ($\mathcal{O}(10^5)$ images). Using automated tools such as ImageJ (and Ilastik?), a large training set can be assembled in just a few hours (!). At prediction time, however, a CNN technique *d-regularly sparse kernels* can be used to process an entire image at once. The study applies this technique on a number of different datasets consisting of a mixture of phase contrast and fluorescence microscopy. They use it also for classifying cell lines in co-culture[21]. When the segmentation masks are produced, a "cellular classification score" is calculated as the majority vote of pixels from each class[22]. As they are most interested in live-cell imaging[23], they detail several pertinent use cases for their technique, including monitoring bacterial cell size over time.

In an ode to reproducible research, the team indicate where their materials can be found online, including a Docker image of their analysis pipeline.

22. Sadanandan, Sajith Kecheril, Petter Ranefall, and Carolina Whlby. "Feature augmented deep neural networks for segmentation of cells." European Conference on Computer Vision. Springer International Publishing, 2016.

This is a dubious paper that claims to extend UNet with "feature augmentation" for segmentation of cells in phase contrast and fluorescence microscopy. They make very modest improvements on the benchmark by creating transformations of the images. They also claim to do "modality transfer learning", but it is very underwhelming: simply redeploying the trained network on a different form of microscopy.

23. Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

This is a relatively technical paper, introducing fully convolutional network for semantic segmentation[24]. Fully convolutional networks (FCNs) are CNNs that continue the convolutions (including possibly deconvolutions) until the end. Thus, the input remains an image (albeit perhaps convolved and downsampled) through to the output. This results in what is referred to as a *dense* prediction. Through a formalism, they demonstrate how all classical CNNs belong to the same family of fully convolutional networks. In classical CNNs, by contrast, the fully-connected layers impose an abstraction of the images into discrete classes.

---

[21] Where two cell lines are studied in the same cell culture.

[22] Ostensibly, that is, as they express this in an inexplicably extravagant way.

[23] Time-lapse microscopy

[24] Semantic segmentation is about localising known objects in an image and segmenting them, whereas segmentation is about recognising general sub-structures such as edges and regions. Depending on the dataset, there may be multiple classes present, and multiple instances of each class. Intuitively, this is a more difficult problem than object classification, or bounding box estimation.

The basis for architecting and training FCNs is to take a pre-trained CNN, such as AlexNet, and reshape its fully connected layers to continue the convolution. Feeding forward on the network will give a heat map of the object in the image, as these will be the neurons (formally fully connected) that are excited when the particular object is revealed (Figure 2). This it the basis for segmentation. The architecture is then extended with an upsampling/deconvolutional layer to give a pixel-to-pixel classification. The authors also experiment with skip architectures, where earlier layers are reused ("fused") alongside higher layers in order to synchronise local and global information,

> Semantic segmentation faces an inherent tension between semantics and location: global information resolves the *what* while local information resolve *where*. (page 1)

The resulting network completes the entire segmentation pipeline (i.e. *in-network*), without resorting to miso-level features such as super-pixels[25] or *post-hoc* analysis. Their architectures are pre-trained ILSVRC models, fine-tuned on various (smaller, $\mathbb{O}(10^3)$ images) datasets, such as PASCAL-VOC[26] and NYUD[27]. The authors use `Caffe` for their experiments, being the lab that originally produced the software, achieving state-of-the-art results.

24. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, Cham, 2015.

This is a very comprehensive paper. It presents u-net, a new fully convolutional architecture, symmetrical in a contraction (downsampling) and expansion (upsampling) path. The network uses skip architectures more extensively than the original FCN paper, concatenating[28] each upsampled layer with the corresponding downsampled layer. They use a pixel-wise softmax in the the log loss function, and combine this with a weight function to assign higher weights to pixels near borders (as these are important and underrepresented in the dataset). The weight function scales by a Gaussian distance. Data augmentation is used extensively (though they do not go into detail), especially because the datasets are very small. U-net succeeds in winning the ISBI cell tracking challenge, and also achieves state-of-the-art (though much worse than human) results on other problems.

[25]Groups of pixels having roughly the same pixel intensities. Forms a tessellation over the image. Can be achieved with any clustering or segmentation algorithm.

[26]Visual Object Classes. PASCAL is a UK-based collaboration.

[27]New York University (NYU) depth dataset. Combines RGB channels with depth information derived from Microsoft Kinect.

[28]Concatenation of layers simply adds layers as additional channels. In u-net, layers are cropped (a curious operation in itself) in order to be compatible.

25. Ciresan, Dan, et al. "Deep neural networks segment neuronal membranes in electron microscopy images." Advances in neural information processing systems. 2012.

This is a good paper that introduces the pixel classification net used for segmentation in the Van Valen paper, rendering that one pedagogically worthless. They use some minor pre- and post-processing techniques to improve their predictions, as well as an ensemble of four nets that, at the time, held first place in the ISBI challenge (ultimately supplanted by u-net and others). This paper was most interesting for its handling of unbalanced datasets. The problem is binary classification (membrane pixel, non-membrane pixel), yet membrane pixels are far less common than non-membrane pixels, though they train on a balanced dataset. So, they compared the network's output probability versus the true probability over the 2.6 million test samples, and found a curve that could be well approximated with a cubic polynomial, which they obtained using least squares regression. The posterior probabilities of the net could therefore be mapped to true probability estimates at inference time, then thresholded at 0.5 in the usual way. The biggest drawback of the pixel classifier method (at the time) is its expensive inference: predicting on tens of thousands of patches per image. As alluded to in Van Valen et al., this shortcoming has been remedied in the interim.

26. Slack, Michael D., et al. "Characterizing heterogeneous cellular responses to perturbations." Proceedings of the National Academy of Sciences 105.49 (2008): 19306-19311.

This paper describes yet another profiling technique. It is rather intuitive, based on clustering a sampled "reference dataset", using features derived from intensity colocalisation (ratio) between each pair of channels at every pixel. The ratios are quantized into a 16-bin histogram, creating a 16×16 histogram for each of the three pairs: (p-p38/DNA, p-ERK/DNA), (DNA/p-p38, p-ERK/p-p38), and (DNA/p-ERK, p-p38/p-ERK) (see supplementary materials). This is repeated for both the nuclear, and membrane areas of each cell. Thus, each cell measures $2 \times 3 \times 16 \times 16 = 1534$ features. After a reduction of to 25 dimensions with PCA, GMMs are used to create a probability profile supported by k sub-populations, and averaged over the cell-population. This choice of supervised learning is quite intuitive,

> [characterising] cell heterogeneity as stochastic variation of feature vectors around a collection of distinct phenotypes. (page 2)

The profiles are used for various analyses, one of which is MOA prediction. The (sensible) choice of distance metric is a symmetrised KL-divergence. The dataset used in the study is the same as in Perlman et al. (2004), but with a few modifications.

27. Boland, Michael V., and Robert F. Murphy. "A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of HeLa cells." Bioinformatics 17.12 (2001): 1213-1223.

This is a seminal (how so?) paper that proposes a neural network classifier for distinguishing 10 categories of organelles in HeLa cells (previously Chinese hamster ovary eggs!). They design several sets of features, some general, and some special-purpose, aimed at characterising expected sub-cellular morphologies. Here, the choice of classifier (feed-forward neural net) is arbitrary (they try several other classifiers)[29]. The analysis is conducted in an astute manner. For example,

> In order to avoid biasing the classification system, the mean and standard deviation of the training data were also used to normalise the stop [read: validation] and test sets. This choice simulates the situation in which a previously trained classifier is applied to data not available at the time of training.

This is something that often overlooked. Dimensionality is reduced using Wilk's λ statistic to assess the ability of features to separate classes. This was found to be more effective than PCA. The study notably resulted in a public available dataset[30].

28. Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.

This is a very famous paper for histogram of oriented gradient (HOG) image features conceived for the problem of person detection. The writing is highly technical and the authors elect to engage in jargon rather than illustrating their technique pictorially. This makes it hard going for the uninitiated, however the technique seems to be the following: the direction of the gradient at each pixel is computed (i.e. the angle of the sum of central difference gradients computed in the x- and y-directions). The directions are summarised as orientation histograms over square groups of pixels called cells, producing a local representation as a vector of proportions (they choose 9 per cell). The cell values are normalised over a wider multi-cellular area called a block. They use this representation to train a linear SVM (or kernelised at the expense of higher runtime). The SVM is deployed on test images as a sliding-window classifier. The outputs of the detector are run through a conventional algorithm, "non-maximum suppression". The pipeline is so successful on the MIT benchmark dataset that they construct their own, more challenging, INRIA person detection dataset.

[29]This work pre-dates the CNN-driven revolution in computer vision and consists of extensive feature engineering, which must go a long way toward capturing the spatial structure of the cell. It is interesting to compare this approach with the raw pixel-based approach of deep learning, where all spatial information is absorbed into the model at once.

[30]https://ome.grc.nia.nih.gov/iicbu2008/hela/index.html

29. Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps." arXiv preprint arXiv:1312.6034 (2013).

This paper presents two techniques for extracting visualisations from trained CNNs. The first aims to synthesise an image that maximally represents a given class, c. This is done by taking the unnormalised neuron for the class, S_c and computing the gradient with respect to an image initialised at zero. This can be used to find I,

$$\arg\max_I S_c(I) - \lambda||I||_2^2$$

Backpropagation is then continued (with trained weights fixed), making incremental "image updates", arriving at a local optimum. This technique produces stunning synthetic images maximising the impression for class c. The images are rich in textures and object components that ostensibly excite the neuronal pathways corresponding to the class' typical image.

The second technique is to produce a saliency map given an image, I_0. Noting that in a linear model, the weights indicate pixel importance directly, the authors consider a first-order Taylor approximation for the score function as a function of the image pixels. The first derivative (the bias term is a throwaway constant) would therefore uniquely indicate those pixels would most change the class score. The saliency map can be constructed as a threshold on the derivatives of greatest magnitude, and can be computed in a single backpropagation! Using this "weakly supervised" segmentation algorithm, the authors devise an object localisation procedure that performs well in the ImageNet challenge.

30. Avondoglio, Dane, et al. "High throughput evaluation of gamma-H2AX." Radiation Oncology 4.1 (2009): 31.

This paper from the National Cancer Institute[31] is a bit wide of the mark. It presents an alternative assay based on electrochemiluminescence for measuring DNA double-strand breaks, induced by radiation[32], of particular interest in radiobiology[33]. They compare their assay with the standard immunofluorescent staining, which they claim is not scalable (at least as of 2009). Other options for detecting DSBs, they state, are immunoblot (involving synthetic proteins) and flow cytometry (involving lasers) assays. They note that γ-H2AX is a trending (as of 2009) marker for DSBs, yet its function is not fully understood. The marker disperses in cells that survive radiation. There is, however, nothing of infomatic interest in the paper.

31. Mining of Massive Datasets

31.1. **Chapter 2 - MapReduce and the New Software Stack.** In contrast with traditional notions of computing power as *supercomputers*, the rise of large-scale Web services

[31]A department of the US National Institutes of Health.

[32]Single-strand breaks (SSB) are (of course) more common.

[33]A field of biology studying the effect of

```
def map(chunk):
    for elem in chunk.elements:
        emit(elem, 1)

def reduce(counts):
    emit(counts['key'], sum(counts['counts'])
```

FIGURE 1. Simple word counting map reduce function.

has brought about *cluster computing*, whereby large quantities of commodity hardware are utilised in parallel. Each processing unit is referred to as a *compute node* and resides in a *rack*. The racks of a cluster are connected by high-speed Gigabit Ethernet cables.

The cluster implements a distributed file system (DFS), such as the original Google File System (GFS) to the open-source Apache Hadoop Distributed File System (HDFS). The system is designed to be robust so that distributed databases are preserved and computing pipelines are maintained when failures occur on individual compute nodes or whole racks. Files are divided into *chunks*, which are replicated in a *quorum* of the cluster. There is a master node that stores a directory of the file system so that files can be located by any node. The master node is further replicated, and all nodes know where this is (presumably according to some consensus algorithm).

MapReduce is a programming framework designed to exploit such a cluster infrastructure. It consists of a master controller that coordinates and distributes the file chunk used in the computation. The framework requires the programmer to write code for a `Map` and a `Reduce` function. Nodes running the `Map` function, *mappers*, prepare (key, value) pairs that are aggregated by key by `Reduce`, running on *reducer* nodes. An example is given in Figure 31.1. In such a case as word count, we could introduce a *combiner* operation to `Map` that aggregates the words in the chunk in advance. This is suitable as the operation of addition is commutative.

MapReduce is suitable for parallel computations on data that is mostly static. If the data is changing all the time, the overhead of replication may be too costly for such a system.

31.2. **Chapter 3 - Finding Similar Items.** For purposes of filtering plagiarism, mirrors, etc. in a search, we need fast algorithms to estimate document similarity.

The idea here is to efficiently estimate the similarity between documents, without exhaustively analysing them in detail. The starting point is to represent a each document in a corpus as a set of all *singles* in the document. We would like to use the Jaccard similarity to compare documents. However, calculating it exactly is infeasible. Therefore, we use create a random permutation of the *characteristic matrix* of the shingle sets. This is a boolean matrix with 1's whenever a shingle is present in the document. It can be easily shown that the probability that two documents have the same shingle is equal to the Jaccard similarity. Therefore, by repeatedly permuting the matrix randomly and comparing the positions of the first 1's, we can *consistently* estimate the Jaccard similarity of two

documents. To simulate the permutations (an expensive operation), we use n randomly chosen hash functions to map rows to rows (this incurs a small, though unimportant error). This creates a *signature* of length n for each document, which can be readily used to calculate the estimated Jaccard similarity. This is called the *minhash* algorithm.

However, we do not wish to calculate the Jaccard distance for all pairs (infeasible). Therefore, we use another hashing technique, *locality-sensitive hashing*, to map blocks of each signature into separate hash spaces. We denote a pair of documents *candidate pairs* if they land in the same bucket for one of these hashes. The choice of block size can be controlled, thereby controlling the probability, giving us a technique for efficiently identifying similar documents at high probability.

31.3. **Chapter 5 - Link Analysis.** This chapter presents PageRank, the algorithm that saved search engines from spam. PageRank models the web as a directed graph of nodes (web sites) and arc (hyperlinks), and aims to assign an importance (and consequently a *rank*) to each node. The importance is the probability of arriving at the page during a random walk[34]. For this, one considers an initial state vector $\mathbf{v}$ of uniform probabilities, and a transition matrix, $\mathbf{M}$. The probability of Because the transition matrix is stochastic, repeated multiplications lead to a steady state where, $\mathbf{v} = \mathbf{M}\mathbf{v}$, that is, an eigenvalue problem. Solving for $\mathbf{v}$ on a graph as large as the web is infeasible, therefore the matrix-vector multiplication is done several times (~ 50 suffices for the entire web).

Note hyperlink information is obtained by web crawlers/spiders[35]. The computation, however, is made by cluster computing grids, perhaps implementing MapReduce.

32. Eulenberg, Philipp, et al. ""Reconstructing cell cycle and disease progression using deep learning." bioRxiv (2017): 081364.

Using a large imaging flow cytometry of individual cells, this study trains an Inception architecture to distinguish cell cycle phases. The penultimate layer (prior to softmax) is visualised with t-SNE in three dimensions, showing a cluster progression in space following the chronology of cell cycle phases $\mathrm{G}_1 \to \mathrm{S} \to \mathrm{G}_2 \to \mathrm{M}$. This is what they refer to as reconstructing cell cycle phase. The result was found to be more accurate and striking than non-deep (boosting) methods.

33. Louppe, Gilles, et al. "Understanding variable importances in forests of randomized trees." Advances in neural information processing systems. 2013.

In this fine paper by Gilles, theoretical results are given for variable importance in ensembles of totally randomised trees[36]. The mean decrease impurity importance of a variable X_m is,

[34]Note that a strategy of assigning importance merely on the *number of links* pointing to a site is vulnerable to spam also (a web site can just link to itself millions of times).

[35]The file `robots.txt` may be included so that the crawler will not index certain pages on a site, but this is only a matter of courtesy and can be ignored.

[36]In such trees, all variables are tested to construct the tree nodes, and in a random order.

$$Imp(X_m) = \frac{1}{N_T} \sum_T \sum_{t \in T : v(s_t) = X_m} p(t) \Delta i(s_t, t),$$

that is, the impurity incurred by splitting on X_m, averaged over all trees and weighted by the probability of arriving at the split for a given tree. Other nice results are given, such as,

$$\sum_{m=0}^{p} Imp(X_m) = I(X_0 \dots X_p; Y),$$

that is, the sum of variable importances is the mutual information between all predictors and the output variable. Most of the results are, however, proven for idealised scenarios where an infinite amount of data and trees are available. The authors further begin to extend the results to random forests, where split variables are chosen by their ability to reduce entropy, not randomly[37]. The paper concludes on a nice toy example of representing digits in a seven-segment display. The seven variables (different horizontal/vertical lines) that construct each digit 'class' can be assessed for their importance.

34. Li, Fei-Fei, Karpathy, Andrej, and Johnson, Justin. "CS231n – Convolutional Neural Networks for Visual Recognition" (2016)

34.1. **Lecture 1 – Introduction.** Lecture 1 is a mild affair. It begins by "tracing" the history of inventions culminating in computer vision. Most notably, the *camera obscura*, whose simple construction embodies the principle underpinning all of photography. This "pinhole camera" focuses light from a source through a small hole on an opaque surface. If the hole is small enough, a ray of light passing through the hole at a given angle corresponds closely to a localised part of the source, resolving a focused image on a light-sensitive material. The material is enclosed in a box to remove ambient light (hence *obscura*). Notice if the hole is cut too large, localisations of the resolved image no longer guarantee a narrow range of angle of incidence, and the output becomes an amalgamation of colours (presumably white light).

Other interesting mentions are normalised cut segmentation (closely related to spectral clustering), and histogram of gradient features.

It is stated that image classification will be the focus of the course, this being one of the most important problems of visual recognition.

34.2. **Lecture 2 – Image Classification Pipeline.** Lecture 2 introduces the image classification pipeline, comprising of *training* and *prediction*.

They note the difficulty of image classification systems: mapping a 3D array of intensity [0, 255] pixels to a semantic class, and all the while robust to changes in viewpoint variation,

[37]Note that while two fully-developed trees may fit the data equally well, decision trees in general are regularised by setting a maximum depth for branching. To minimise generalisation error within this quota, variables should be chosen that most decrease impurity. This is where random forests really differ from totally randomised trees, and where the theoretical results diverge.

illumination, deformation, occlusion, background clutter, and general intraclass variation. There is simply no way to program this explicitly. There must rather be a data-driven approach, where classifications are based on probability instead of rigid certainty. In this sense, machine learning can be seen as a form of non-deterministic programming, where problems are solved by maximising a likelihood model over data samples[38].

The first classifier they present is nearest-neighbour/kNN (k odd), having the uncommon property of a short training time and long prediction time. These they use with with L1 (manhattan) or L2 (Euclidean) distances. This is for illustrative purposes only, however. They emphasise that NN classifiers are never used on images–the high dimensionality makes it impossible. Remember, however, NN becomes perfectly accurate as the training data tends to the set of all possibly images. For CIFAR-10 this is $(3x256)^{(}32x32)$ images. A parametric approach sacrifices this ultimate accuracy, aiming to generalise sufficiently with a parameterised model of fixed size.

34.3. **Lecture 3 – Loss Functions and Optimisation.** Lecture 3 rounds out the discussion on linear models. Multiclass linear SVMs and multinomial logistic regression are covered with explanations of hinge loss and softmax. L_2 regularisation is introduced as a means of preferring many smaller weights than few larger weights. They summarise the linear modes as having a) a score function ($f(\mathbf{x}) = \mathbf{W}x$), and b) a loss function. They summarise with a very nice diagram.

Next, they motivate the optimisation procedure, starting with a naive random search. Then, they try calculating the numerical gradient, but note its excessive computation time and inferiority as an approximation to the analytic gradient (vector of partial derivatives). They do, however, emphasise the use of a numeric gradient for checking analytic correctness (the *gradient check*). They formalise gradient descent, then mini-batch GD (usually 32, 64, 128, 256 samples to benefit from vector-based computations).

The final slides summarise some traditional feature extraction algorithms: Colour histograms, HOG/SIFT, visual bag of words[39]. These are finally contrasted with CNNs (in-built feature extractor).

34.4. **Lecture 4 – Backpropagation and Neural Networks.** Lecture 4 introduces backpropagation as an efficient algorithm for computing the partial derivatives of the weights of different layers w.r.t. the loss function at a given iteration of training. They conceptualise it with a network diagram, expressing the (partially) ordered operations required to compute an output function, $f(x, y, z)$. Given a neural network is a DAG embodying nested computations, a derivative computed at any node will be a factor in all immediately preceding nodes, as a consequence of the chain rule. Thus, the computation of any derivative involves only $\mathcal{O}(N)$ computations for N graph nodes. Notice that backpropagation reduces trivially in "shallow" models.

[38]A recent blog post by Andrej Karpathy makes the same characterisation: https://medium.com/@karpathy/software-2-0-a64152b37c35

[39]This is interesting. It seems to consist of specifying a number of useful pixel arrangements as words in a visual "dictionary", then using kNN to classify each component of a new image as a visual word, giving a final bag-of-words representation as in NLP.

Recall, the (sub-)gradient of the max function, $\max(x, y)$ is the indicator function $\mathbb{1}(x, y)$, since,

$$f(x, y) = \max(x, y) = \begin{cases} x & x \geq y \\ y & y > x \end{cases}$$
$$\implies \frac{df}{dx} = 1 \begin{cases} x & x \geq y \\ y & y > x \end{cases}$$
$$\implies \frac{df}{dx} = 1(x \geq y)$$

There are some miscellaneous points to complete the lecture:

- Surprisingly, a feed-forward net is trivial to implement in `NumPy` (indeed, with any reliable linear algebra library).
- There is a loose inspiration for artificial neural nets in the neural connectivity of the human brain. Brain cells (neurons) receive signals via tree-like dendrites, and propagate signals via an axon, which ultimately branches out itself into axon terminals and interfaces with the dendrites of other neurons with a synapse.
- In general, more neural nodes corresponds with more capacity, but regularisation should be controlled by other means than architecture size (penalties, dropout).

34.5. **Lecture 5 – Training Neural Networks, Part 1.** Lecture 5 summarises the nitty-gritty details of training neural networks. It begins with activation functions:

- Sigmoid: classical, but saturated (high-valued) neurons return a value very close to 1, and a gradient very close to 0 i.e. $d\sigma/dx = \sigma(x)(1 - \sigma(x)) \approx 0$ when x is large and $\sigma(x) \approx 1$. This even happens quite quickly in practice. Another problem is that the sigmoid is not zero-centered (centered on an output of 0.5).
- tanh: zero-centers data (linear transformation of the sigmoid), but has the same saturation problem.
- ReLU (rectified linear unit): does not saturate (at least not in the positive direction), is more computationally efficient the previous two, and tends to converge in fewer iterations. However, it is not zero-centered and ReLU units *die*: if the outputs are always negative, the gradient is always 0 and the weights are never updated. This could happen e.g. if the weights were primarily negative and the input was always a strong positive. From an intuitive perspective, this could happen if the input found prominent features in every training sample and a codependency arose between certain neurons in the network, leading to an equilbrium where certain neurons always fired positive, and certain proceeding neurons were always negative.
- Leaky RELU: fixes the cell death problem with ReLU $\triangleq f(x) = \max(0.01x, x)$. There are many variants on this including parametric ReLU (PReLU), where $f(x) = \max(ax, x)$ and a is an updated weight. There are also ELU (exponential

linear units), where the activation is linear for positive x, exponential for negative x.
- Maxout: Takes max between two competing linear weights.

ReLU is most recommended, then the variants, then tanh. Sigmoid is to be avoided.

They proceed to discuss data preprocessing. Zero-centering and normalising the data is essential, but PCA, decorrelation (diagonal covariance), whitening (identity covariance) are not common in deep learning. Typically, the mean image or per-channel means are subtracted.

Weight initialisation seeks to ensure weights are well distributed to promote successful training. The classical notion of initialising to small, random values is reasonable for small networks, but not for deep neural nets. It can be seen that the activations approach zero as we propagate inputs through layers. An improvement to this is Xavier normalisation, which gives quasi-Gaussian activations. This is recommended as an intialisation strategy. A state-of-the-art technique is batch normalisation (BN), which is an additional, trainable layer. A BN layer normalises the features of an input layer over the current batch of samples. It provides the network with additional learnable parameters to undo the normalisation if optimal. BN is performed prior to activation. Some training tips:

- Once initialised, a sanity check is to propagate all training data once through the model. The loss (without regularisation) ought to be $\ln(C)$ for C classes.
- Another sanity check is to overfit on a small part of the training data.
- No improvement to loss $\implies$ learning rate too low; NaN $\implies$ learning rate too high
- No improvement then major improvement $\implies$ bad initialisation
- Disparate train vs. validation accuracy $\implies$ increase regularisation; equal train vs. validation accuracy =¿ increase network capacity.
- Random hyperparameter search can outperform grid search. There is at least some intuitive sense to this: if the loss "terrain" varied randomly and discontinuously, there would be no strategy with higher expectation than any other. However, neural net loss terrain is a partially smooth function. Randomly chosen points therefore maximise the expectation, but a grid search may not! Note also that grid search wastes time on unimportant parameters e.g. we might try $reg = 1\text{e}{-3}$ with $lr = 1\text{e}{-5}, 1\text{e}{-4}, 1\text{e}{-3}$, giving the same each time (lr unimportant). A random search would have tried 3 different regulrisation parameters in the same time.

Miscellaneous:

- The `Caffe` model zoo is a GitHub Wiki page indexing available pre-trained models for `Caffe`. These are bundled in a standardised format.
- Interesting one-line if/else for assignment in `Python`

34.6. **Lecture 6 – Training Neural Networks, Part 2.** Lecture 6 continues the discussion on training neural nets, focusing primarily on optimsation algorithms, batch gradient descent with momentum:

```
# update weights
```

```
w = w - lr * dw

# update weights with momentum
w = w - lr * dw + mu * v
# update velocity
v = mu * v - lr * dw
```

Thus, when we add the momentum term, we are adding part of the previous gradient step, e.g. 0.5, a smaller part of the one before that, and even smaller part of the one before that, etc. Thus, if we are making small, but consistent progress in one dimension, and oscillating around another, the momentum will "build up" in the progressive dimension, and dampen in the other direction:

$$du + 0.5 \times du + 0.5^2 \times du + ...$$

vs.

$$dv - 0.5 \times du + 0.5^2 \times du - ...$$

- The Nesterov momentum update does the same, but evaluates the gradient *after* accounting for velocity.
- Adagrad keeps track (cache) of the sum of squares of gradient updates, and divides the update step by the square root of this sum.
- RMSProp does the same, but adds a decay rate to the cache updates.
- Adam (most recommended) is like the above but with yet more parameters.

In addition to all this, the learning rate must be scheduled to decay.

They next talk about dropout, which approximates an ensemble of exponentially many nets. The key point to remember is that the dropout rate, p, at each layer of activation decreases the expectation by a factor of p. Hence, at test time, each activation layer must be scaled up by p. An alternative is to scale down the activations by p at train time (inverted dropout). This removes the need for scaling at test time.

They finally comment on some historical curiosities: the Hubel and Wiesel experiments from the 50s, which demonstrated the firing of neurons (captured as pulses of sound) when exposing a stunned cat to shades and shapes on a projector screen. This told us a lot about the function of the visual cortex, including that cells assemble to capture visual features. When we look at the activations in a ConvNet, we see the bright areas that are activated at each layer. We can think of the kernels (groups of weights) as learned feature templates. They are high when they convolve with a close approximation to the learned template. Thus, a visualisation of each of the activation maps shows the parts of the image that excite the neurons, layer by layer: first edges, later objects, finally abstractions.

34.7. **Lecture 7 – Convolutional Neural Networks.** Lecture 7 examines CNNs in detail. This has been examined substantially elsewhere so will not be covered here. Generally it does not bring up anything new. ZF-net might be worth a look at some point (or, then again, maybe not). Some points to remember:

- Zero-padding is used to allow convolutions on every input pixel (including the border), an ensure an output of like size.
- To count the number of parameters in a convolutional layer, compute $(S \times S \times D + 1) \times F$, where S is the filter size, D is the depth of the input layer, and F is the number of filters (usually a power of 2). Notice we add 1 for bias.

34.8. **Lecture 8 – Spacial Localisation and Detection.** Lecture 8 expands image classification to localisation, detection, and segmentation.

In object localisation, we return the bounding box (x, y, width, height) tuple for associated classified object. We typically assess the performance with the Jaccard distance on the pixel sets. This is trained as a regression problem with L_2 loss. One obvious way of achieving this is to take a pre-trained net and replace the "classification head" with a "regression head". Localising multiple objects, for example, in human pose estimation, could have multiple bounding box outputs. Another approach is to do this with a sliding window.

In object detection, we are faced with a quandary. Whereas in localisation, we were concerned with associating a bounding box one-to-one with a singular object, here we may have any number of objects (included repetitions of the same class) in a single image. Regression therefore does not cut it. We can, however, apply classification at many different positions and scales. Another approach is with "region proposals". R-CNN (regions with CNN) is a framework for object detection. Here, a pre-trained CNN is fine-tuned on positive-negative examples of region proposals of objects from a number of classes. The features of these are saved to disk (!). Post-hoc classifiers (SVM) are trained on the feature set (one per object class). Also, a regression model is trained to adjust the bounding box (outputs tuple of offsets). Fast R-CNN, faster R-CNN are improvements to the original framework.

34.9. **Lecture 9 – Understanding and Visualising Convolutional Neural Networks.** Lecture 9 presents ideas for visualising aspects of CNNs. These include:

- visualising the weights (Gabor-like filters)
- sampling FC layers (CNN codes), plotting them in t-SNE space
- generating images: "Deconv" approaches (ZF-net), backprop on image from output neurons, inverting CNN codes, Google DeepDream, Neural Style.

The Simonyan paper seems to be very important. This has been summarised elsewhere. This was improved upon in a subsequent paper. Deep Dream seems to derive from this, though the blog post does not reveal much. Neural style trains over a new loss composed of content + style loss functions. All use pre-trained CNNs to work their magic, and effectively reverse the process of image recognition.

34.10. **Lecture 10 – Recurrent Neural Networks.** Lecture 10 comes at long last to Recurrent Neural Networks (RNN). Whereas with other models a single vector goes in and a single vector comes out, in an RNN, a sequence of vectors goes in and/or comes out. The internal hidden states are organised in a Markov chain.

In its most basic form, the hidden state is the tanh activation of the weighted previous hidden state and weighted input. The output is the weighted new hidden state.

RNNs can usually be trained using backpropagation through time (BPTT), an algorithm whereby the recursive structure is "unraveled" and backpropped as a feed forward network.

They then show how RNNs can be used to sample all sorts of nonsense. The nonsense has structure, however, and is superficially meaningful.

One very nice application is for image captioning. Here, we have a training set consisting of many captioned images. First, a CNN is trained for image classification. Then, an RNN is trained on the corresponding captions where the initial hidden state is the final fully connected layer of the CNN! To borrow a phrase from Hinton, this CNN code represents a "thought". Presumably, though the CNN was trained on simple classes and not image captions, this CNN code representation gives an amalgamation of object information detected in the image. This is what helps the RNN to give a rich description. At inference time, the RNN is "seeded" with the CNN code of a test image, and a `<start>` token. The output is fed back in as input iteratively until the RNN emits the `<end>` token.

In LSTM (long short term memory), each hidden unit, h, receives a memory cell, c. This is computed by element-wise combinations of the previous memory cell and hidden state. The purpose of the memory units are to address the vanishing gradient problem due to repeated multiplications of a weight matrix with spectral radius ¡ 1 (if the spectral radius is ¿ 1, the gradient explodes, but this has a simpler solution in *clipping*). It would be excellent to read the original paper on LSTM one day.

34.11. **Lecture 11 – CNNs in Practice.** Lecture 11 covers practical aspects of training/using CNNs

Begins by talking about data augmentation. They note how widely used it is, and list horizontal flips, random crops, and color jitter. When noise is added, at test time the noise is somehow marginalised, as it is in dropout. They then present two transfer learning ideas:

(1) freeze all layers prior to softmax and use the CNN as a feature extractor for a linear model. This can be used for the same, or related problems as that which trained the CNN.
(2) freeze earlier layers only, fine tune (with a much smaller learning rate $\sim 1\%$ of original). When a lot of data is available, this can be used. The number of layers fine-tuned depends on the similarity of the dataset; only low-level features (early layers) will be relevant when the datasets are quite different.

They further point out that transfer of CNN codes are used everywhere in computer vision problems.

Then, in a change of focus, they describe the benefits of using the smallest possible convolutional kernel: 3×3. It was shown in the VGG paper that three 3×3 kernels were more efficient to compute than a single 7×7 kernel, despite having the same receptive field size.

They make a point on using 1×1 filters... but it's not clear what they're saying. This somehow relates to the inception blocks of GoogLeNet.

Again changing direction, they explore how convolutions can be rewritten as matrix multiplications, by reshaping receptive field and kernel tensors to vectors. Though this is expensive memory-wise, we can then benefit from the fast matrix multiplication of linear algebra libraries such as `BLAS` (used by `NumPy`). There is finally a discussion of using Fast Fourier Transforms to do the convolutions.

Next there is a note on hardware: `NVIDIA` GPUs much more common than AMD.

They also discuss using lower precision to speed up computation e.g. *half* precision (half of single, quarter of double). Experiments have been done with "binary nets" also (1-bit precision).

34.12. **Lecture 12 – Software Packages: `Caffe`/Torch/`Theano`/`TensorFlow`.** Lecture 12 focuses on the software packages available for doing deep learning

- Beginning with `Caffe`, which notably has bindings for `MATLAB` and `Python`. Blobs are for storing data and derivatives. `Caffe` is designed to avoid writing code (net defined in prototxt etc.). Funnily enough, `Caffe` operates on HDF5 files! The Model Zoo is a repository for architectures on GitHub. It is not good for RNNs or large networks.
- `Torch` is was developed at NYU and IDIAP! and written in C and `Lua`! Interfaces easily with the GPU, while preserving `NumPy`-like usability. Models specified by writing `Lua` code (except `PyTorch`, presumably), with many predefined modules available.
- `Theano` comes from Bengio's lab in Montreal. Inputs are defined symbolically to build up a computational graph, similar to `TensorFlow`. It has a higher-level API, Lasagne. Also `Keras`, which additionally interfaces with `TensorFlow`. Lasagne has a model zoo.
- `TensorFlow` is very similar to `Theano`. `TensorFlow` is overall very good, with a very nice visualisation tool. Few pretrained models as of Feb 2016 (very likely to have changed). Best at RNNs!

The lecture ends with an appendix on characteristics of `Caffe`

34.13. **Lecture 13 – Segmentation and Attention.** Lecture 13 picks up where Lecture 8 left off, exploring the final mode: segmentation

For this, they contrast two varieties: semantic segmentation (pixel-wise), and instance segmentation (simultaneous detection and segmentation).

Semantic segmentation (Ciresan etc.) is quite straightforward: extract patch, classify central pixel, continue. Other approaches to semantic segmentation use fully convolutional nets... multi-scaled, aggregating outputs, or simply upsampling the final layers (this is FCN) with skip connections (aggregations).

Instance segmentation has similar pipelines to object detection like fast-RCNN. Not so intellectually interesting. They go on to talk about soft/hard attention models. This is lost on me.

34.14. **Lecture 14 – Videos and Unsupervised Learning.** Lecture 14 is the much anticipated final lecture on video and unsupervised learning!

They begin by describing 3DCNNs for processing video data. Also, schemes involving RNNs to model longer-term dependencies. This is, again, not so interesting to me.

Finally they pass to unsupervised learning, beginning with auto-encoders. They note that former wisdom saw such deep networks as requiring special training procedures in order to avoid poor local minima: pre-training piece by piece, then fine-tuning. Today, however, what with ReLU, BatchNorm, ADAM etc., training is straightforward.

They then talk about variational auto-encoders and generative adverserial networks (GANs), which both merit their own time, so will not be summarised here.

35. Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).

In the auto-encoding variational Bayes model, one postulates the existence of a set of latent factors, $\mathbf{z}$ generating observations $\mathbf{x}^{(i)}$ as part of a random process. One begins with the log-probability of the marginal distribution on the observed i.i.d data,

$$\log p(\mathbf{X}) = \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)})$$

where the parameters θ are the true parameters of this generative model. The method proposes to deal with a *recognition model*, $q_\phi(\mathbf{z}|\mathbf{x})$, with parameters ϕ, that approximates the true, generative posterior, $p_\phi(\mathbf{z}|\mathbf{x})$, which suffers from the typical problem of intractibility. With this in mind, consider the log-probability of a single observation,

$$\begin{aligned}
\log p_\theta(\mathbf{x}^{(i)}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}^{(i)})\big] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})} \cdot \frac{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})}\right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right] \\
&= \underbrace{D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})\big)}_{\geq 0} + \mathcal{L}(\theta,\phi;\mathbf{x}^{(i)})
\end{aligned}$$

where $\mathcal{L}(\theta,\phi;\mathbf{x}^{(i)}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}^{(i)},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}^{(i)})\big]$. Given $D_{KL}(\cdot) \geq 0$, we can instead consider maximising, $\mathcal{L}(\theta,\phi;\mathbf{x}^{(i)}) \leq \log p_\theta(\mathbf{x}^{(i)})$, as this maximises a lower bound on the log probability. This is what is called the variational lower bound on the marginal likelihood, and is the key to variational models.

A rearrangement of this likelihood that will prove to be useful is,

$$\mathcal{L}(\theta,\phi;\mathbf{x}^{(i)}) = -D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})\big) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\big]$$

Let us position this milestone. We are attempting to fit parameters by maximising the likelihood of the observed data. We have supposed there is an underlying process of latent variables $\mathbf{z}$ generating these observations. We have adopted a generative framework[40] and replaced the problematic posterior distribution with a recognition model $q_\phi(\mathbf{z}|\mathbf{x})$. We have used the positivity property of KL divergence to eliminate the posterior and to work with a lower bound to the marginal likelihood.

Now, we would like to maximise this bound through differentiation. The typical approach to is to approximate with Monte Carlo techniques:

$$\begin{aligned}
\frac{\partial}{\partial\phi}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[f(\mathbf{z})\big] &= \frac{\partial}{\partial\phi}\int q_\phi(\mathbf{z}|\mathbf{x})f(\mathbf{z})d\mathbf{z} \\
&= \int \frac{\partial}{\partial\phi}q_\phi(\mathbf{z}|\mathbf{x})f(\mathbf{z})d\mathbf{z} \\
&= \int q'_\phi(\mathbf{z}|\mathbf{x})f(\mathbf{z})d\mathbf{z} \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[f(\mathbf{z})\frac{\partial}{\partial\phi}\log q_\phi(\mathbf{z}|\mathbf{x})\big] \\
&\approx \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z})\frac{\partial}{\partial\phi}\log q_\phi(\mathbf{z}^{(l)}|\mathbf{x})
\end{aligned}$$

for samples $\mathbf{z}^{(l)}$. The penultimate step follows from the fact that $\frac{\partial}{\partial\phi}\log q_\phi(\mathbf{z}^{(l)}|\mathbf{x}) = q'_\phi(\mathbf{z}|\mathbf{x})/q_\phi(\mathbf{z}|\mathbf{x})$. The problem with this approach is that it exhibits high variance (we must draw many samples).

To remedy this, the authors come up with a novelty called the *reparameterisation trick*. This consists of replacing the random variable $\mathbf{z}$ with a deterministic function of $\mathbf{x}$ and a random noise variable $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ such that,

$$\mathbf{z} = g(\mathbf{x}, \boldsymbol{\epsilon})$$

The function $g(\mathbf{x}, \boldsymbol{\epsilon})$ is chosen to be differentiable (in our case an *encoding* network). Now we have the *stochastic gradient variational Bayes* (SGVB) estimator,

$$\widetilde{\mathcal{L}}(\theta, \phi; \mathbf{x}^{(i)}) = \underbrace{-D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})\big)}_{\text{analytic solution}} + \underbrace{\frac{1}{L}\sum_{l=1}^{L}\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})}_{\text{Monte Carlo approximation}}$$

[40]Generative models are desirable for many reasons. Strong assumptions are made on the model parameters by assigning prior and likelihood distributions. This has a powerful regularising effect on the model. By contrast, discriminative models fit the posterior "blindly", and rely on regularisation techniques to avoid catastrophic overfitting. Discriminative models have the advantage of allowing arbitrary feature transformations. This quickly makes generative models intractable.

where $\mathbf{z} = g(\mathbf{x}, \boldsymbol{\epsilon})$ and $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$. This now suggests an gradient ascent algorithm. Optimisation may be done in *mini-batches*, sampled randomly from the full dataset, $\mathbf{X}$. We therefore optimise $\widetilde{\mathcal{L}}(\theta, \phi; \mathbf{X}^M) = \sum_{i=1}^{M} \widetilde{\mathcal{L}}(\theta, \phi; \mathbf{x}^{(i)})$ for minibatch size M. The authors thus specify the *auto-encoding variational Bayes*[41] algorithm, based on gradient ascent. The authors find that for sufficient M, the Monte Carlo approximation may be done with a single sample ($L = 1$).

It remains to choose distributions for the probabilities. In a *variational auto-encoder* (VAE), we chose the following:

(1) $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. This is apt since by this we are only assuming the latent variables are emitted independently, and according to parsimonious assumptions. The form is convenient, as it is conjugate to the recognition model.

(2) $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_E, \sigma_E^2 \mathbf{I})$. Hence, the covariance matrix is diagonal. Here E denotes encoder. The mean and variance are computed by an encoder network in the following way:

$$\begin{aligned}
\mathbf{h}_E &= \tanh(\mathbf{W}_E^{(1)} \mathbf{x} + \mathbf{b}_E^{(1)}) \\
\boldsymbol{\mu}_E &= \mathbf{W}_E^{(2)} \mathbf{h}_E + \mathbf{b}_E^{(2)} \\
\boldsymbol{\sigma}_E^2 &= \exp(\mathbf{W}_E^{(3)} \mathbf{h}_E + \mathbf{b}_E^{(3)})
\end{aligned}$$

As a result, $\boldsymbol{\phi} = \{\mathbf{W}_E^{(1)}, \mathbf{W}_E^{(2)}, \mathbf{W}_E^{(3)}, \mathbf{b}_E^{(1)}, \mathbf{b}_E^{(2)}, \mathbf{b}_E^{(3)}\}$. As the two above two distributions are involved in a divergence term in the estimator, they may be integrated analytically. The resulting expressing (in terms of the Gaussian parameters) is differentiable with respect to the neural parameters, $\boldsymbol{\phi}$. Hence, in the gradient ascent of the AEVB algorithm, this part is straightforward.

(3) $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_D, \sigma_D^2 \mathbf{I})$. In this case, the Gaussian parameters come from a decoder network. The parameters are specified in a similar way to the above, with the addition of a sampling step:

$$\begin{aligned}
\mathbf{z}^{(l)} &= \boldsymbol{\mu}_E + \boldsymbol{\sigma}_E \odot \boldsymbol{\epsilon}^{(l)} \\
\mathbf{h}_D &= \tanh(\mathbf{W}_D^{(1)} \mathbf{z}^{(l)} + \mathbf{b}_D^{(1)}) \\
\boldsymbol{\mu}_D &= \mathbf{W}_D^{(2)} \mathbf{h}_D + \mathbf{b}_D^{(2)} \\
\boldsymbol{\sigma}_D^2 &= \exp(\mathbf{W}_D^{(3)} \mathbf{h}_D + \mathbf{b}_D^{(3)})
\end{aligned}$$

where $\boldsymbol{\epsilon}^{(l)}$ is sampled from $\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$. As a result, $\boldsymbol{\theta} = \{\mathbf{W}_D^{(1)}, \mathbf{W}_D^{(2)}, \mathbf{W}_D^{(3)}, \mathbf{b}_D^{(1)}, \mathbf{b}_D^{(2)}, \mathbf{b}_D^{(3)}\}$. Since the loss function uses the log of this distribution, this yields a differentiable function of the parameters $\boldsymbol{\theta}$.

[41] The eponymous algorithm has a name that can be confusing. At this point in the derivation, traditional auto-encoders are nowhere to be seen. However, the implied graphical structure of latent factor model, and the encoding/decoding distributions invoked by the conditionals are what give the model its name

This derivation is sufficient to train the model. One fascinating use of VAEs is the way they may be used to generate data, most spectacularly images. One may choose different dimensionalities for the latent space of $\mathbf{z}$. Given that each coordinate in the latent space is chosen to be independent (the prior on $\mathbf{z}$ has diagonal covariance), one can sample each coordinate separately. Therefore, one may choose a probability for each of the dimensions of $\mathbf{z}$, feed it into the inverse Normal CDF and obtain a coordinate. The full vector of coordinates can then be fed through the decoder network to obtain the mean $\boldsymbol{\mu}_D$. Reshaping this mean to the right dimensions yields a synthetic image. For a two-dimensional latent space, one may choose equally spaced probabilities on the unit square (i.e. probabilities between 0 and 1). One can then visualise the entire latent manifold.

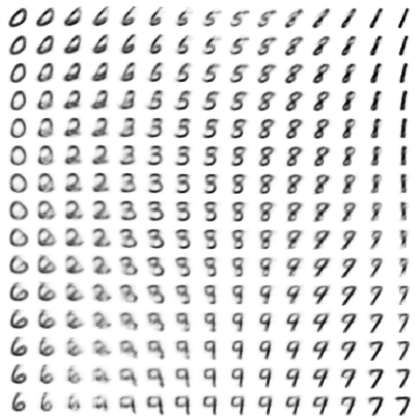


FIGURE 2. Plot of two-dimensional manifold. Produced with `fchollet/keras/examples/variational_autoencoder_deconv.py`

36. GANIN, Y., USTINOVA, E., AJAKAN, H., GERMAIN, P., LAROCHELLE, H., LAVIOLETTE, F., ... & LEMPITSKY, V. (2016). DOMAIN-ADVERSARIAL TRAINING OF NEURAL NETWORKS. JOURNAL OF MACHINE LEARNING RESEARCH, 17(59), 1-35.

This longer work from 2016 combines the theoretical underpinnings and original conception of domain-adversarial neural networks (DANNs) (Ajakan et al., 2014) with their extension to deep architectures (Ganin, Lempitsky, 2015).

We are here concerned with unsupervised domain adaptation. That is, domain adaptation between source $S = \{(\mathbf{x}_i, y_i)\} \sim \mathcal{D}_S$ and target $T = \{\mathbf{x}_i\} \sim \mathcal{D}_T$ datasets, noting the target is *unlabelled* (unsupervised), or partially labelled (semi-supervised).

The inspiration for DANNs lies in work done by Ben-David et al. in "A theory of learning from different domains" (2006, 2010), where they defined the $\mathcal{H}$-divergence,

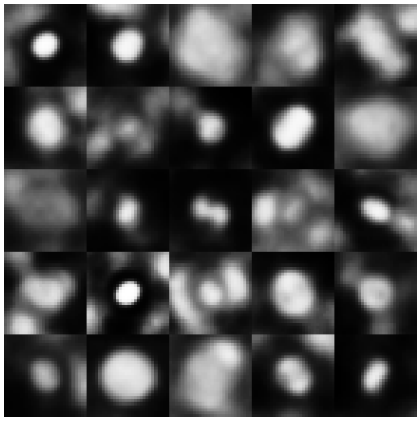
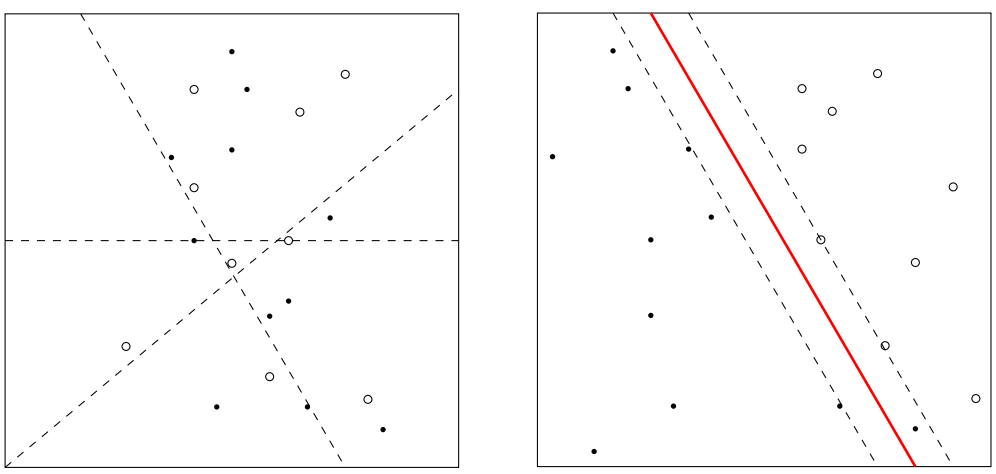


FIGURE 3. Random samples from VAE trained on (downsampled) annotated morphological cell line screen data (20 dimensional latent space).

$$d_{\mathcal{H}}(D_S^X, D_T^X) = 2 \sup_{h \in \mathcal{H}} \left| P_{\mathbf{x} \sim D_S^X}\big(h(\mathbf{x}) = 1\big) - P_{\mathbf{x} \sim D_T^X}\big(h(\mathbf{x}) = 1\big) \right|$$



(A) Low divergence (B) High divergence

That is, given a source domain (distribution), D_S^X (marginalised by the input variable), and a target domain D_T^X, and given a hypothesis class[42] $\mathcal{H}$, the divergence between the source and target domains with respect to the $\mathcal{H}$ is the classifier (here binary–for simplicity), that, proportionally, most classifies the domains into separate classes.

[42]Category of classifiers, e.g. linear SVMs.

To illustrate, suppose we choose our hypothesis class to be a linear SVM. This class is capable (through choice of the SVM weights) of creating any possible linear hyperplane. Suppose the two domains were linearly separate in the feature space. It would then be possible to train an SVM to perfectly separate the two, putting one domain fully into the positive class, and the other into the negative class. This would maximise the inner expression of the divergence formula, thus describing a high divergence. Should we have, on the other hand, two highly overlapping domains, any hyperplane cut through the middle of the point cloud would give us a divergence close to 0. Should we cut so as to isolate a few outlier points of a particular class, these would represent only a small proportion of the data, and the probability would also be close to 0. Intuitively, the $\mathcal{H}$-divergence captures the effect nicely.

Conveniently, it is possible to compute a consistent estimate of the $\mathcal{H}$-divergence with finite data with the *empirical* $\mathcal{H}$-divergence,

$$\hat{d}_{\mathcal{H}}(S,T) = 2\left(1 - \min_{h\in\mathcal{H}}\left[\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}[h(\mathbf{x}_i)=0] + \frac{1}{n'}\sum_{i=n+1}^{N}\mathbb{1}[h(\mathbf{x}_i)=1]\right]\right)$$

for samples, $S \sim D_S^X$ of size n and $T \sim D_T^X$ of size n'. For convenience, these are indexed from $1 \to n$ and $n+1 \to N$, where $N = n + n'$. Though this may be hard to do compute precisely in general, we can approximate this simply by training a classifier of the class $\mathcal{H}$ on the constructed dataset, $U = \{(\mathbf{x}, 0) : \mathbf{x} \in S\} \cup \{(\mathbf{x}, 1) : \mathbf{x} \in T\}$, that is, a classifier to differentiate between the domains. The estimated generalisation error, ϵ of this classifier could then be used to approximate the empirical domain divergence as $2(1 - 2\epsilon)$.

They first formalise a neural network as a learning model of the form,

$$E(\theta_f, \theta_d, \theta_y) = \left[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(G_y(G_f(\mathbf{x};\theta_f);\theta_y), y_i) + \lambda R(\theta_f, \theta_d)\right]$$

where the G_f component acts as a feature extractor, and G_y is a classifier. These may have any neural architecture, shallow, deep, convolutional, etc.

Recalling that the target data is unlabelled, the authors propose to define the regulariser of their model as,

$$R(\theta_f, \theta_d) = -\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d^i(\theta_d, d_i) - \frac{1}{n'}\sum_{i=n+1}^{N}\mathcal{L}_d^i(\theta_d, d_i)$$

where $\mathcal{L}_d^i(\theta_d, d^i) = \mathcal{L}(G_d(G_f(\theta_f);\theta_d), d_i), d_i \in \{0, 1\}$. Thus formulated, we have a loss maximising the negative of the minimisation term in the divergence formula. Therefore, adding this regulariser to the objective function, and maximising the objective with respect to its parameters, θ_d, give a worse overall loss. At the same time, however, the feature parameters, θ_f, which are trained to minimise the objective, are chosen to minimise this maximisation of the regulariser by the divergence parameters, θ_d, thus maximising the minimising component of the divergence, thereby minimising the divergence. The feature parameters therefore play in dual role in both minimising classification loss, and minimising

divergence (equivalently, maximising discrimination error), *adversarially* to the domain classifier. This promotes domain invariant features.

Formally, we have,

$$(\theta_f, \theta_y) = \arg\min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d)$$
$$\theta_d = \arg\max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d)$$

which can be found as a saddle point of the update steps,

$$\theta_f \leftarrow \theta_f - \mu\left(\frac{\partial \mathcal{L}_y^i}{\partial \theta_f} - \lambda \frac{\partial \mathcal{L}_d^i}{\partial \theta_f}\right)$$
$$\theta_y \leftarrow \theta_f - \mu \frac{\partial \mathcal{L}_y^i}{\partial \theta_y}$$
$$\theta_d \leftarrow \theta_f - \mu \frac{\partial \mathcal{L}_d^i}{\partial \theta_d}$$

Note that θ_f is updated to minimise the classification loss, but maximise the domain classification loss. This is called the *adversarial* step. There is a nice visualisation of all this in Figure 1.

The contribution of Ganin and Lempitsky is to apply this to deeper architectures (in particular, convolutional), and to introduce a *gradient reversal layer* to enable a smooth implementation work without altering implementations of gradient descent (this still assumes that the local gradients can be specified arbitrarily). This is achieved with the pseudo-function,

$$R_\lambda(\mathbf{x}) = \mathbf{x}$$

$$\frac{\partial R_\lambda}{\partial \mathbf{x}} = -\lambda \mathbf{I}$$

This is inserted between the feature extractor and the domain classifier in the model formulation. It should be emphasised that this is not a mathematical trick, but rather a programming hack. The gradient reversal layer is easy to implement and has been done in the in Caffe (by the authors), TensorFlow, etc.

Results are shown first for the shallow architecture of the original paper. It is peculiar that, though the DANN prevails in most categories, it does so only marginally, and the authors use a statistical test (!) to verify that it outperforms the vanilla models by a substantive amount. This tests are done on the Amazon reviews dataset, a task of sentiment analysis transfer between reviews of different categories of product.

The deep DANN results are far more impressive, with adaptation done between different digits datasets. For example, MNIST to M-MNIST (modified), which is built from a random crop of natural images, followed be inverting the pixels corresponding to the

original M-NIST digits. Another notable dataset used is the Street View House Number (SVHN) dataset. The OFFICE dataset (office appliances) is also used. Another task is person reidentification.

37. Frosst, N., & Hinton, G. (2017). Distilling a Neural Network Into a Soft Decision Tree. arXiv preprint arXiv:1711.09784.

This paper describes a technique for improving the accuracy of a soft decision tree by using a pre-trained neural network. A soft decision tree is a hierarchy of filters, where the decision probabilities are the output of a logistic function with learnable weights. Note, in a classical decision tree, a single feature is considered at a time. In a soft decision tree, the full sample vector is used at once. These filters direct observations down to a final leaf layer of *bigots* (as opposed to experts[43]), which learn a *static* (bigoted) probabilities distribution over the classes. That is, a distribution independent of the input. The ground truth is the *target* distribution, which are normally simply one-hot encodings of the true classes. When this one-hot target distribution is replaced (or augmented) with the output probabilities from a pre-trained neural network, this is called *distillation*. This leads to a small performance increase. Thus, the soft targets help isolate cases that are difficult to classify by the (more powerful) neural network. These corner cases are able to be redirected to different parts of the tree. Thus, information about the relative difficulty of classifying a particular sample is instilled into the training set. N.B. this paper is not a method to transform a neural network into a decision tree, rather to transfer knowledge about the targets.

Integrating high-content screening and ligand-target prediction to identify mechanism of action

38. Young, Daniel W., et al. "Integrating high-content screening and ligand-target prediction to identify mechanism of action." Nature chemical biology 4.1 (2008): 59-68.

This presents another (and the last of the five reviewed in Ljosa et al., 2013) phenotypic profiling technique. This is in the form of a factor model, which decomposes the per-cell feature matrix into 6 phenotypic factors (decided by eigenanalysis). Though the model is barely explained in the body (the supporting materials are not available on Nature's website), it allows a grouping of orthogonal features. The averaged factors for a well (treatment) produce a profile that is used in various ways. The most interesting (though slightly dubious) is matching the clustering of these profiles (which reveals seven major groups) with the clustering of the perturbation molecules, where the similarity measure used Tanimoto (Jaccard?) similarity. The paper is nice to look at but devoid of mathematical interest. The abundance of molecular diagrams explain why it was published in Nature Chemical Biology. There are a few nice quotes, however,

[43] A mixture of experts is an ensemble method that learns gating functions (filters) that decide which *expert* (weak learner) to use. This can be arranged in a tree-like hierarchy.

> Compounds with similar structure have similar function, and quantitative structure-activity relationships (SARs) are at the heart of drug discovery. (page 4)
>
> Dealing with complexities of predictive toxicology will require breakthroughs in cytological image analysis, target prediction schemes and data mining. (page 9)

39. Horbach, S. P., & Halffman, W. (2017). The ghosts of HeLa: How cell line misidentification contaminates the scientific literature. PloS one, 12(10), e0186281.

This paper studies the proliferation of misidentified cell lines in scientific literature. The paper gives a nice summary on the origins of cell lines, and how contaminated (erroneous) primary research passes to secondary research. The authors use the International Cell Line Authentication Committee (ICLAC) database[44] to source their research, identifying 451 cell lines leading to 32755 primary papers, and an estimated half a million secondary papers. Their methodology erred on the side of caution and can therefore be considered conservative. They further demonstrate that the problem is global, and not particular to regions with a less respected research tradition. The problem of contaminated research also continues unabated to the present day, despite the decades-long awareness of the problem and improved cell line verification techniques such as *short tandem repeats* (STR). There are some valid counter-arguments that the authors address, yet the vastness of the problem remains, and many researchers appear to be unaware they are studying misidentified cell lines. The authors leave their recommendations in the final section. Note that this problem is additional to the problem of genetic drift in cell lines.

40. Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." Nature 323.6088 (1986): 533-538.

In this historic paper, Hinton et al. present backpropagation for training neural networks. This approach had a long history prior, but this, and the results within popularised neural networks in the late 80s. The crux of the algorithm is the formula,

$$\frac{\partial E}{\partial y_j} = \sum_i \frac{\partial E}{\partial x_i} w_{ij}$$

That is, for a neuron y_j, its gradient with respect to the loss function E, is the sum of the gradients of neurons emanating from it times the corresponding weights.

> The most obvious drawback of the learning procedure is that the error-surface may contain local minima so that gradient descent is not guaranteed to find a global minimum. However, experience with many tasks shows that the network very rarely gets stuck in poor local minima that are significantly

[44] Thankfully, neither MDA231 nor MDA468 are present in this database.

> worse than the global minimum. We have only encountered this undesirable behaviour in networks that have just enough connections to perform the task. Adding a few more connections creates extra dimensions in weight-space and these dimensions provide paths around the barriers that create poor local minima in the lower dimensional subspaces. (page 2)

More than thirty years on, in 2017, Hinton advocates a renunciation of backpropagation. This is reflected in the closing remark,

> The learning procedure, in its current form, is not a plausible model of learning in brains. (page 3)

41. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International Conference on Machine Learning. 2015.

This is really impressive stuff. Batch normalisation (BN) considers the *internal covariate shift* of a neural net as the main reason neural network training has traditionally been so difficult. Unlike in a linear model, the intermediate layers of a neural network receive inputs whose distribution changes over time, that is, as the weights of preceding layers are updated. This leads to erratic behaviour as these distributional fluctuations are propagated to deeper layers. This explanation is assuredly close to the truth as batch normalisation is an apparent panacea, acting as a powerful regulariser, allowing much learning rates, leading to vastly more rapid convergence and even higher accuracy. It further renders dropout obsolete (it performs better without it) and reduces the importance of weight decay. The paper achieves state-of-the-art performance in training (one variant of) the Inception network, and world record accuracy on ImageNet with an ensemble of batch normalised nets.

Batch normalisation aims to approximate the benefit of *whitening* data[45]. Incorporating this into a network could be very costly, however. In the buildup to BN, the authors show how naively normalising without incorporating the normalising components into the gradient update would lead to a fixed output and loss, while updated the parameters indefinitely. BN therefore seeks to achieve a fixed/stable mean and variance rather than a fixed distribution. Therefore, BN consists of the steps, $\boldsymbol{\mu}_B = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i$, that is, the column- or feature-wise average, and $\boldsymbol{\sigma}_B^2 = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \boldsymbol{\mu}_B)^2$, where $\mathbf{x}_i$ are the batch activations from the previous layer. These are combined to form the normalised,

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \boldsymbol{\mu}_B}{\sqrt{\boldsymbol{\mu}_B + \epsilon}},$$

for ϵ small. In a final step, BN introduces learnable parameters γ and β to create $\hat{\mathbf{y}} = \gamma\hat{\mathbf{y}} + \beta$ and restore the representational power, that is, the normalisation may be

[45]Given matrx $\mathbf{X}$, we can calculate $\boldsymbol{\Sigma} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$, the covariance matrix. The eigen-decomposition of this (which, perhaps confusingly, we might obtain using SVD) can be written $\boldsymbol{\Sigma} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$. The *decorrelated* matrix $\hat{\mathbf{X}} = \mathbf{X}\mathbf{U}$ has a diagonal covariance matrix, $\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\mathbf{U}^T\mathbf{X}^T\mathbf{X}\mathbf{U} = \mathbf{U}^T\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T\mathbf{U} = \boldsymbol{\Lambda}$. Furthermore multiplying by $\boldsymbol{\Lambda}^{-1/2}$ will render the covariance matrix an identity.

undone if necessary. These then become a part of the backpropagation. The vector $\hat{\mathbf{y}}$ is then fed through the activation function. When convolutions are concerned, the batch normalisation is a little different and this is described by the authors.

42. Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

Generative adversarial nets (GANs) are the realisation of a framework whose stated aim is to synthesise images, by sampling from the distribution learnt by the model. They rival variational autoencoders for image synthesis[46]. It is generative in the sense of approximating the pdf of the likelihood on the data. The framework is adversarial as it posits a pair of models in a two-player minimax *game* (as in game theory). One model, the discriminator aims to minimise the error in discerning true data from that sampled from the second model, the generator, which aims to maximise the error made by the discriminator,

$$\min_D \max_G V(D,G) = \frac{1}{m}\sum_{i=1}^{m} \log D(\mathbf{x}^{(i)}) + \log(1 - D(G(\mathbf{z}^{(i)})))$$

where $\mathbf{x}^{(i)}$ is the (unlabelled) training data $\mathbf{z}^{(i)}$ are sampled random noise. In this form, the underlying game is *zero-sum*[47]. The models are usually neural nets, indeed deep and convolutional. The optimum is a *Nash equilibrium* (a saddle point) between the two objectives. It may be shown that the distribution implicitly defined[48] by the generator G, $p_g = p_{\text{data}}$ at optimality, and $D = p_{\text{data}}/(p_{\text{data}} + p_g)$ where p_{data} is the true distribution behind the data. Training is done by backpropagation and gradient descent in pairs. Since this paper was published in 2014, many improvements have been found for GANs, as shown in (Goodfellow, Ian. "NIPS 2016 tutorial: Generative adversarial networks." arXiv preprint arXiv:1701.00160 (2016).). There are some impressive image synthesis results from GANs. One nice example is iGAN, a software develop to assist image manipulation. This is capable of generating photo-realistic images according to some simple brush-stroke prompts from the user. Another fascinating demonstration interpolates between two distinct objects by taking steps across the manifold. Each intermediate sample is a perfect example of the semantic class, that magically eventuates in form of the target object, showing the model has learned the space of all 'realistic' entities from that class.

[46]VAEs are not known to be universal approximators, and appear not to generate samples of such high quality, but remain far easier to train.

[47]If we reformulate the objective to be two separate minimisations, each is the negative of the other

[48]GANs are implicit generative models, in contrast to explicit models such as VAEs, which model the probabilities directly.

43. Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size. arXiv preprint arXiv:1602.07360.

In this highly affable paper, the authors present the SqueezeNet family of architectures, based on a general-purpose micro- and macro-architectural design strategy for neural networks. The core of SqueezeNets are a module (micro-architecture) called the *fire module*. This consists of a *squeeze* layer of 1×1 filters, followed by an *expand* layer combining 1×1 and 3×3 filters. This approach embodies a set of three design principles aimed at reducing the number of model parameters while preserving good (AlexNet-level) accuracy. The design is instructive, and elucidates some of the confusing details around the design of the Inception model. For example, the pseudo-function of aggregating outputs from differently-sized filters. Given the 1×1 and 3×3 filters of the expand layers produce the same activation (they use the same stride and the 3×3 are zero-padded), there is mystery to be borne. Another nice flourish is global average pooling[49], eschewing the need for affine layers. The authors try different ratios of number of filters between squeeze and expand layers, as well as within expand layer. They also experiment with ResNet-style bypass connections. The final, chosen model (the simple bypass variant of SqueezeNet with tuned filter ratios) achieves or exceeds AlexNet accuracy with a 500-fold reduction in number of parameters! This makes it feasible to store a deep neural net on an embedded chip.

44. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

In another splendid paper from Kingma *Adam* (adaptive moment estimation), a first-order gradient descent algorithm, is presented. Adam incorporates a self-annealed choice of step size based on estimates of the gradient's element-wise expected value and normalised by the (uncentered) element-wise variance. These are approximated by a running exponential average with a bias correction. The ratio of mean and root variance estimates represent a sort of *signal-to-noise ratio* for adapting the learning rate on the fly. The authors present a convergence analysis on the algorithm's regret bound,

$$R(T) = \sum_{t=1}^{T} |f_t(\theta^t) - f_t(\theta^*)|,$$

where the functions f_t are the distinct cost functions invokes by a randomly sampled mini-batch at each iteration (the algorithm is conceptualised as optimising an online, stochastic process). The optimal parameters are θ^*. They succeed in showing that $R(T)/T = \mathcal{O}(1/\sqrt{(T)})$ in a lengthy proof in the appendix. Surprisingly, the mathematics to establish this are just a simple (yet tedious) series of arithmetic manipulations. The

[49]Average pooling is an alternative to the traditional max pooling, in which square blocks are averaged to create the downsampling. *Global* average pooling averages over the entire activation map, returning a single element per filter, which is then passed into a softmax layer. Thus, there is a direct connection between output classes and the final layer of activation maps.

read is quite gratifying and instructive. They further show connections to RMSprop and AdaGrad, and show show replacing the variance with higher moments leads to a nicely simplified algorithm for the limit in ∞, which they call *AdaMax*. They naturally show state-of-the-art results in optimising various classifiers.

45. Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. 2010.

This important paper dates back to the neural net renaissance, just prior to the watershed moment in 2012. The analysis of the paper presents two main findings: firstly, that the sigmoid function is unsuitable for deep networks due to a saturation effect that may take an infeasibly long time to escape (and with inferior outcomes). They visualise the distribution of parameters and gradients per layers over a large number of epochs, illustrating the saturation effect nicely. They speculate that early in learning, the network can often make small, "easy" gains using the biases alone, pushing the weights to zero, and the corresponding gradients as a result. For this reason, they advocate zero-centered, symmetric activation functions such as the hyperbolic tangent tanh() (itself a linearly transformed sigmoid), or the then recently proposed *softsign* function, whose short-lived successes were soon eclipsed by the rectified linear unit, and apparently forgotten to deep learning history. Nevertheless, these sigmoid alternatives have similar (if slightly less severe) problems.

Interestingly, they note, all activations work better when a net's weights are initialised to those of a pre-trained autoencoder of like architecture. Note that this was the fleeting moment in deep learning history in which deep neural nets were not trained from scratch. Autoencoders have the useful property of allowing a piece-wise greedy[50] approach, layer by layer, fixing each layer's weights in turn. Without end-to-end training, the autoencoder will tend to be a sub-optimal fit, yet still provides a good initialisation to a corresponding neural classifier, which is subsequently fine-tuned. Note that this is no longer standard practice, although learning transfer between nets follows the same idea.

With some analysis, they formulate the dynamics of weight and activation variance to be exponential in the number of layers, with respect to $nVar(W_i)$, with n the number of neurons in each of the adjacent layers. This is why an earlier proposed standard initialisation, $W_{ij} \sim U(-1/\sqrt{n}, 1/\sqrt{n})$, which results in $nVar(W_{ij}) = 1/3$ leads to "vanishing" gradients in deep layers. The authors therefore propose an initialisation distribution having the property, $nVar(W_{ij}) = 1$, ensuring approximately equal variance at each layer. This they achieve with their *normalised initialisation*,

$$W_{ij} \sim U\left(\frac{-\sqrt{6}}{\sqrt{n_i + n_j}}, \frac{\sqrt{6}}{\sqrt{n_i + n_j}}\right)$$

where n_i and n_j are the incoming and outgoing layer sizes of the weights, combined in a harmonic mean–a compromise between potentially different layer sizes. This yields

[50]Greedy algorithms yield an approximate solution by performing a local optimisation at each step.

$nVar(W_i) \approx 1$[51]. This initialisation strategy came to be known as Xavier initialisation (or, more correctly, *Glorot* initialisation). The authors show on three image datasets (including their own Shapes-2×3 dataset) that this initialisation more of less consistently improves test error (although still fall short of a pre-trained model).

46. Gordon, Geoff, and Ryan Tibshirani. "Gradient descent revisited." Optimization 10.725/36 (2012): 725.

This is a presentation about gradient descent and related concepts. It is quite a rich introduction. The centerpiece is a convergence proof for gradient descent. The reasoning can be hard to follow as there are space restrictions in the slide format. For this reason, the proof is repeated here with more intermediate steps.

Firstly, suppose we have a convex function f whose gradient f' is Lipschitz continuous with constant M. As a result, the second derivative is bounded by M. Consider the Taylor expansion of f at a point y, in one dimension for brevity:

$$f(y) = f(x) + f'(x)(y - x) + R_1(y)$$

where

$$R_1(y) = \int_x^y f''(t)(y - t)dt \leq \int_x^y M(y - t)dt = \frac{M}{2}(y - x)^2$$

Note that in higher dimensions, M is chosen such that $M\mathbf{I} \succeq \nabla^2 f(\mathbf{x})$, that is, $M\mathbf{I} - \nabla^2 f(\mathbf{x})$ is negative semi-definite[52]. Thus, in general we have,

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) + \frac{M}{2}||\mathbf{y} - \mathbf{x}||_2^2$$

Consider a gradient step: $\mathbf{x}^+ = \mathbf{x} - t\nabla f(\mathbf{x})$. By substitution, we have,

$$\begin{aligned} f(\mathbf{x}^+) &\leq f(\mathbf{x}) + f'(\mathbf{x})(\mathbf{x}^+ - \mathbf{x}) + \frac{M}{2}||\mathbf{x}^+ - \mathbf{x}||_2^2 \\ &= f(\mathbf{x}) - t\nabla f(\mathbf{x})^T \nabla f(\mathbf{x}) + \frac{Mt^2}{2} f(\mathbf{x})^T \nabla f(\mathbf{x}) \\ &= f(\mathbf{x}) - t \cdot \left(1 - \frac{Mt}{2}\right) \cdot ||\nabla f(\mathbf{x})||_2^2 \end{aligned}$$

Note that if we choose $0 < t \leq 1/L$, we can guarantee convergence since then,

$$f(\mathbf{x}^+) \leq f(\mathbf{x}) - \frac{t}{2}||\nabla f(\mathbf{x})||_2^2$$

Next consider the optimal point x^*. Since f is convex, we can establish the relation,

$$f(\mathbf{x}) \leq f(\mathbf{x}^*) + \nabla f(\mathbf{x})^T(\mathbf{x}^* - \mathbf{x})$$

[51]Recall the variance of a uniformly distributed random variable, $X \sim U(a, b)$ is $(b - a)^2/12$.

[52]The symbol $\succ$ reads "succeeds", and refers to order. In linear algebra, $\mathbf{A} > 0$ means that $\mathbf{A}$ is positive-definite

and the objective function decreases after each iteration. Combining these inequalities gives,

$$f(\mathbf{x}^+) \leq f(\mathbf{x}^*) + \nabla f(\mathbf{x})^T(\mathbf{x}^* - \mathbf{x}) - \frac{t}{2}||\nabla f(\mathbf{x})||_2^2$$

Then, we have,

$$\begin{aligned}
f(\mathbf{x}^+) - f(\mathbf{x}^*) &\leq \frac{1}{2t}\big(2t\nabla f(\mathbf{x})^T(\mathbf{x} - \mathbf{x}^*) - t^2||\nabla f(\mathbf{x})||_2^2\big) \\
&= \frac{1}{2t}\Big(2t\nabla f(\mathbf{x})^T(\mathbf{x} - \mathbf{x}^*) - t^2||\nabla f(\mathbf{x})||_2^2 - ||\mathbf{x} - \mathbf{x}^*||_2^2 + ||\mathbf{x} - \mathbf{x}^*||_2^2\Big) \\
&= \frac{1}{2t}\Big(||\mathbf{x} - \mathbf{x}^*||_2^2 - ||\mathbf{x} - t\nabla f(\mathbf{x})^T - \mathbf{x}^*||_2^2\Big) \\
&= \frac{1}{2t}\Big(||\mathbf{x} - \mathbf{x}^*||_2^2 - ||\mathbf{x}^+ - \mathbf{x}^*||_2^2\Big)
\end{aligned}$$

Now, summing to iteration k gives us,

$$\begin{aligned}
\sum_{i=1}^{k} f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*) &\leq \sum_{i=1}^{k} \frac{1}{2t}\Big(||\mathbf{x}^{(i-1)} - \mathbf{x}^*||_2^2 - ||\mathbf{x}^{(i)} - \mathbf{x}^*||_2^2\Big) \\
&= \frac{1}{2t}\Big(||\mathbf{x}^{(0)} - \mathbf{x}^*||_2^2 - \cancelto{0}{||\mathbf{x}^{(1)} - \mathbf{x}^*||_2^2 + ||\mathbf{x}^{(1)} - \mathbf{x}^*||_2^2} + \dots\Big) \\
&= \frac{1}{2t}\Big(||\mathbf{x}^{(0)} - \mathbf{x}^*||_2^2 - ||\mathbf{x}^{(k)} - \mathbf{x}^*||_2^2\Big) \\
&\leq \frac{1}{2t}||\mathbf{x}^{(0)} - \mathbf{x}^*||_2^2
\end{aligned}$$

We know from before that $f(x^{(k)}$ is non-increasing. Therefore,

$$f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*) \leq \frac{1}{k}\sum_{i=1}^{k} f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*) \leq \frac{||\mathbf{x}^{(0)} - \mathbf{x}^*||_2^2}{2tk}$$

which concludes the proof. We therefore see there is a convergence rate of $\mathcal{O}(1/k)$.

47. Schönenberger, Felix, et al. "Discrimination of cell cycle phases in PCNA-immunolabeled cells." BMC bioinformatics 16.1 (2015): 180.

This is a real garden variety paper. The team propose some variations on a very standard pipeline for assigning cell cycle phenotypes (G_1, early, mid, and late S, and G_2 phases) using proliferating cell nuclear antigen (PCNA) as a biomarker. They claim they have optimised it for both their confocal and widefield data, though no real tuning is done. One informative paragraph details the approach to multiclass SVMS: training multiple one-vs-many and using Hamming distance to assign classes to test data. Otherwise disappointing.

48. Otsu, Nobuyuki. "A threshold selection method from gray-level histograms." IEEE transactions on systems, man, and cybernetics 9.1 (1979): 62-66.

This humble correspondence (approaching 30,000 Google Scholar citations) proposes a now classic technique for unsupervised thresholding, what has come to be known as *Otsu thresholding*. The technique optimises the placement of a threshold k in a histogram. Noting that the technique was motivated for image segmentation, it is general-purpose, non-parametric approach to thresholding (a form of one-dimensional clustering), it may be used for any univariate feature readout.

The *discriminant* idea is to maximise the ratio $\lambda = \sigma_B^2/\sigma_W^2$ of the *intra*-class variance[53], σ_W^2 and the *inter*-class variance[54], σ_B^2, creating a threshold that maximises similarity within the respective clusters, while maximising the distance between them. The problem is simplified, however, by noting the relation $\sigma_B^2 + \sigma_W^2 = \sigma_T^2$, where σ_T^2 is the (fixed) global variance. It can then be shown that maximising λ is the same as maximising $\eta = \sigma_B^2/\sigma_T^2$, as they are a function of one another (the problem has only one degree of freedom, not two). Since σ_T^2 is a constant we can optimise the threshold merely by maximising σ_B^2. So, the algorithm proceeds by varying k and calculating the inter-class variance each time, retaining the largest.

One may remark also upon the high quality of writing in this article, with a precision that is rare in the contemporary literature.

49. Rose, France, et al. "Compound Functional Prediction Using Multiple Unrelated Morphological Profiling Assays." SLAS TECHNOLOGY: Translating Life Sciences Innovation (2017): 2472630317740831.

This slightly confused paper introduces (for the first time?) a way of predicting small compound mechanism of action by pooling phenotypic effects across multiple cell lines. It is based on the methodology described in the Ljosa review from 2013. The large, rich dataset used were the combined screens of 9 malignant pleural mesothelioma cell lines (supplied by Biophenics). The cell lines were treated with the same panel of 1200 drugs, the Prestwick library, and imaged with a single fluorescent stain–Hoechst. For these, 193 nuclear features were extracted per cell, and averaged into a per well profile vector (further averaged over two replicates). The Prestwick set was categorised into MOA using ChemBank and DrugBank. Approximately half the compounds had a unique MOA and were therefore unsuitable for the leave-one-out cross validation method.

The standard BBBC021 dataset was additionally used. In this case, 453 features were extracted over three fluorescent channels. This seems to have been used to validate the Ljosa methodology (leave-one-out validation with 1-NN based on cosine distance), demonstrating that the MOA prediction accuracy increases as channels are added to the primary (DAPI). Note this was 38 compounds with 2-3 replicates giving 103 conditions with 12

[53]Intra-class variance is the weighted sum of the respective variances of classes invoked by a choice of k

[54]Inter-class variance is merely the average squared distance of the class means from the global mean

MOAs. The best accuracy is then 0.83. They make the rather careless assertion, however, that the ascent from 0.68 accuracy (DAPI alone) to merely 0.83 (all channels combined) calls into question the information gained from the other channels.

To predict MOA across cell lines, one random forest classifier was trained to predict MOA from averaged feature profile on each assay (cell line). To make an ensemble prediction, the probabilities from each random forest are summed together, with the maximum chosen as the prediction.

From this, however, it is not clear how significant the results are. They compare the relative performances of the Ljosa method (rather than comparing the methods directly): 0.83 accuracy on 12 MOAs (0.083 random guessing) vs. 0.10 (top-1) accuracy for 113 MOAs (0.09 random guessing) (using all cell lines). Nor is it clear to what extent the respective cell line morphologies are being leveraged (for example, 10 cell lines correspond to 10 times more data anyway).

50. Bubeck, Sébastien. "Convex optimization: Algorithms and complexity." Foundations and Trends in Machine Learning 8.3-4 (2015): 231-357.

50.1. **Chapter 1 - Introduction.** This introductory chapter introduces the fundamental concepts studied throughout the book, beginning with the convex set[55] The key moment is the statement of the *separation theorem*. This theorem affirms the existence of a hyperplane (creating two *half-spaces*) separating a convex set $\mathcal{X}$ and any point x_0 outside the convex set. Clearly, a non-convex set does not have this property. This is, of course, the cornerstone of machine learning. The theorem is possibly more easily understood in terms of the function: $f(\mathbf{x}) = \mathbf{wx} + t$. Such a function implies a hyperplane for which $f(\mathbf{x}) = 0$. It is easy to show that this hyperplane "decision boundary" runs perpendicular to the direction of vector $\mathbf{w}$, meeting at a point t units from the origin. It may also easily be shown that all points $\mathbf{x}$ above the hyperplane satisfy $f(\mathbf{x}) > 0$ and those below satisfy $f(\mathbf{x}) < 0$. The theorem asserts that if $\mathcal{X}$ is convex, at least one separating hyperplane exists.

A related theorem is the *supporting hyperplane* theorem. This states that a point x_0 on the boundary of a (closed) convex set has a hyperplane that is tangential to the set. A convex set is said to *closed* when it has a boundary, that is, the supremum of the the set. The rest of the set is referred to as the *interior*. An interior is the set of points for which there exists a *ball* neighbourhood of non-zero radius also in the set.

Another concept is the epigraph (points above the graph of a function). The convexity of a function's epigraph is equivalent to the convexity of a function.

The remainder of the chapter gives an overview of the five chapters chapter of the book, outlining a highly appetising bill of optimisation methods to come. Finally, the concept of the "black box model" is proferred: a point of optimisation analysis making the minimum of assumptions about the function.

[55]Some generalisations of convexity are *star convex* sets, holding the convex property for at least one point in the set (think of the center of a geometric star), and further generalising the connected components.

51. Elkan, Charles. "The foundations of cost-sensitive learning." International joint conference on artificial intelligence. Vol. 17. No. 1. Lawrence Erlbaum Associates Ltd, 2001.

This paper is something of a classic. It begins by setting out the general framework for a decision rule given classifier posteriors:

$$i^* = \arg\min_i \sum_j p(j|x)c(i,j)$$

where i^* is the class selected, $p(j|x)$ is the probability of class j given observation x and $c(i,j)$ is the cost of deciding i when the actual outcome is j. It may be, then, that the optimal risk-minimising decision is not the most probable outcome, if the cost of an error is high enough. In the binary case, we obtain the threshold p^* on a decision as,

$$p^* = \frac{c(1,0) + c(0,0)}{c(1,0) - c(0,0) + c(0,1) - c(1,1)}$$

In most machine learning problems, the cost of a correct classification is zero, and indeed misclassification error is symmetric. In such a simple case, $p^* = 0.5$ (as usual).

There is some evidence (though no consensus) that machine learning algorithms perform better when datasets are balanced. This leads to strategies for balancing class representation by (uniformly) *undersampling* the majority class, oversampling (bootstrapping) the minority class (equivalent to reweighting the cost function), or data interpolation algorithms like SMOTE, or data augmentation for images. In the case of undersampling, we could create a balanced dataset. In such a case, the change in priors, $p'(y)$ and $p(y)$ (approximately equal to the respective proportions of classes in the sampled and non-sampled datasets), leads to bias in the posterior probabilities $p'(y|x)$ and $p(y|x)$. Under the assumption that the class conditional probabilities are the same $p'(x|y) = p(x|y)$, it is possible to derive a relation using Bayes theorem,

$$p' = b' \frac{p - pb}{b - pb + b'p - bb'}$$

where p and p' are the uncorrected and bias-corrected posteriors, and b and b' are training and testing class priors (what Elkan calls the *base rates*). One would likely choose $b = 0.5$ for training, in which case,

$$p' = \frac{p}{p + (1-p)(1-b')/b'}$$

s From this result, Elkan shows how a the new prior (level of subsampling) should be chosen such that the adjusted probabilities match,

$$n' = n \frac{p(1-p')}{p'(1-p)}$$

52. Dal Pozzolo, Andrea, et al. "Calibrating probability with undersampling for unbalanced classification." Computational Intelligence, 2015 IEEE Symposium Series on. IEEE, 2015.

Continuing from the Elkan paper, the authors of this paper articulate a full undersampling strategy:

(1) Balance classes (50-50) by multiplying by $\beta = p(+)/p(-) \approx N^+/N^-$.
(2) Train the model.
(3) Adjust posteriors p_s to,
$$p' = \frac{\beta p_s}{\beta p_s - p_s + 1}$$
(4) Adjust the threshold τ_s by solving Erkan's Theorem 1 retroactively (fixing β) for,
$$\tau' = \frac{\beta \tau_s}{(\beta - 1)\tau_s + 1}$$
For $\tau_s = 0.5$, the formula simplifies to,
$$\tau_s = \frac{\beta}{\beta + 1}$$

On occasion, the authors are simply fearless:

> [...] the lower the BS the more accurate are the probabilistic predictions of a model. (page 6)

53. Bengio, Yoshua. "The consciousness prior." arXiv preprint arXiv:1709.08568 (2017).

Purest, unadulterated Bengio is unleashed in this short paper, which proposes a framework for emulating conscious reasoning in neural networks. This is facilitated by a structural constraint on a deep recurrent neural net known as the *consciousness prior*,

$$c_t = C(h_t, c_{t-1}, z_t)$$

where c_t is the abstract "thought" at time t, derived from the previous thought c_{t-1}, a hidden-dimensional hidden state h_t (representing the contents of the network's "brain" at time t), and a random noise vector z_t (emulating the musing of one thought to the next), combined through a function C, being an attention function on some low-dimensional projection of the hidden state. The layer C and the deep network behind it implied by h_t is called the *consciousness RNN*. Bengio further proposes a *verified network* to ensure the coherence between previous hidden states and future thoughts.

It would be interesting to know if such fanciful papers have precedent in Bengio's oeuvre, or indeed in the machine learning literature, and whether such an idea will be pursued, as, by its nature, the paper has the appearance of a wild stab in the dark (nevertheless, Bengio does offer guidance on how to build a proof of concept).