Supplementary material

Data curation and preprocessing

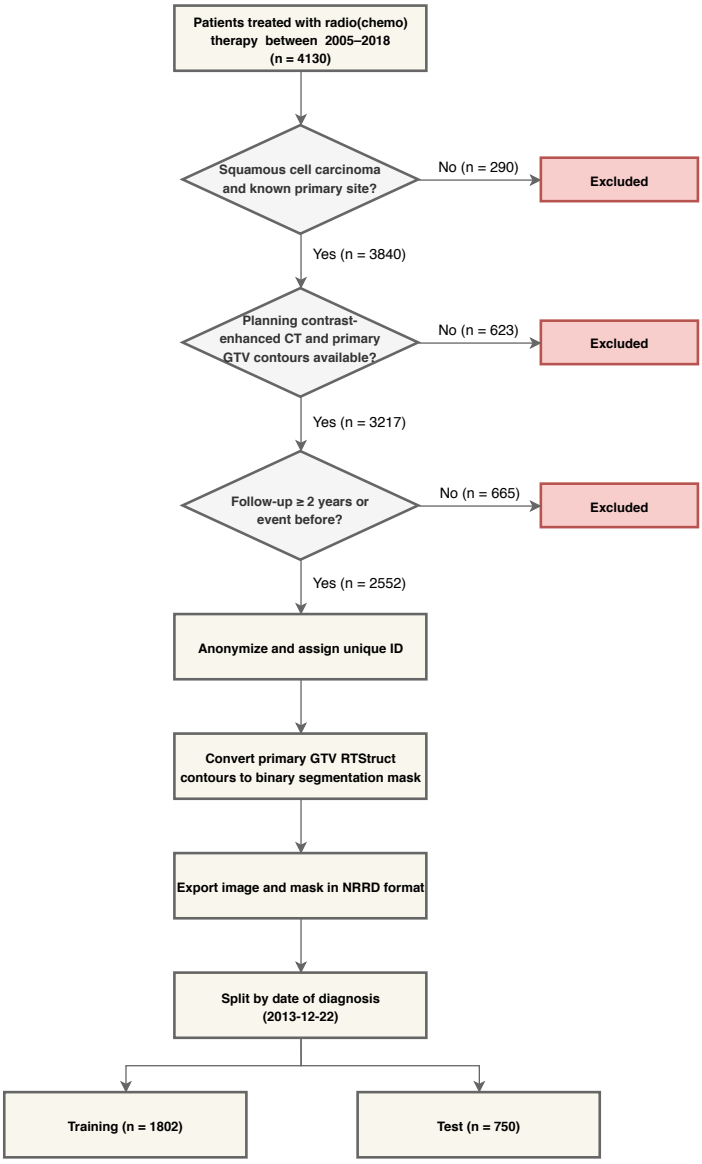

Figure S1. Patient selection and data curation process.

Patient characteristics

	Training	Test
# of patients	1802	750
Outcome		
Alive/Censored	1065 (59%)	609 (81%)
Dead	737 (41%)	141 (19%)
Dead before 2 years	323 (18%)	103 (14%)
Sex		
Male	1424 (79%)	615 (82%)
Female	378 (21%)	135 (18%)
Disease Site		
Oropharynx	777 (43%)	399 (53%)
Larynx	555 (31%)	172 (23%)
Nasopharynx	220 (12%)	101 (13%)
Hypopharynx	115 (6.4%)	28 (3.7%)
Lip / Oral Cavity	72 (4.0%)	10 (1.3%)
Nasal Cavity	31 (1.7%)	20 (2.7%)
Paranasal Sinus	16 (0.9%)	10 (1.3%)
Esophagus	14 (0.8%)	8 (1.1%)
Salivary Glands	2 (0.1%)	2 (0.3%)
T stage		
1/2	919 (51%)	405 (54%)
3/4	855 (47%)	336 (45%)
Not available	28 (2%)	9 (1%)
N stage		
0	688 (38%)	226 (30%)
1	170 (9%)	90 (12%)
2	835 (46%)	395 (53%)
3	108 (6%)	39 (5%)
Not available	1 (<1%)	0 (0%)
AJCC stage		
I/II	455 (25%)	158 (21%)
III/IV	1318 (73%)	581 (77%)
Not available	29 (2%)	11 (1%)
HPV status		
Positive	513 (28%)	327 (44%)
Negative	269 (15%)	168 (22%)
Not tested	1020 (57%)	255 (34%)
Chemotherapy		
Yes	725 (40%)	361 (48%)
No	1077 (60%)	389 (52%)
ECOG performance status		
0	1120 (62%)	436 (58%)
1	489 (27%)	290 (39%)
2	145 (8%)	20 (3%)
>2	34 (2%)	4 (1%)
Not available	14 (1%)	0 (0%)

Table S1. Patient characteristics in the training and test sets.

Imaging protocols

Parameter	Median [range]
Slice thickness	2 [2–2.5]
kVp	120 [120–120]
Tube current [mAs]	300 [121–540]
Pixel spacing [mm]	(.976, .976) [(.702, .702)–(1.17, 1.17)]

Table S1. CT imaging parameters used in the study. Images were acquired using either of: Toshiba Aquilion One ($n = 1653$), GE Discovery ST ($n = 643$), GE LightSpeed Plus ($n = 130$), Philips Brilliance Big Bore ($n = 96$) or GE Discovery 610 ($n = 30$).

Detailed descriptions of submissions

Submission 1

Multi-task logistic regression (MTLR) uses a sequence of dependent regressors to predict the probability of event occuring at multiple discrete timepoints in a multi-task fashion[1]. The model is trained by minimizing the MTLR log-likelihood:

$$
\begin{aligned}
L(\boldsymbol{\Theta}, D) = & \sum_{j=1}^{N-N_c-1} \sum_{k=1}^{K-1} (\boldsymbol{\theta}_k^T \phi(\mathbf{x})^{(j)} + b_k) y_k^{(j)} && \text{(Uncensored)} \\
& + \sum_{j=N-N_c}^{N} \log\Big(\sum_{i=1}^{K-1} \mathbf{1}\{t_i \geq T_c^{(j)}\} \exp\Big(\sum_{k=i}^{K-1} ((\boldsymbol{\theta}_k^T \phi(\mathbf{x})^{(j)} + b_k) y_k^{(j)}) \Big) \Big) && \text{(Censored)} \\
& - \sum_{j=1}^{N} \log\Big(\sum_{i=1}^{K} \exp\Big(\sum_{k=i}^{K-1} \boldsymbol{\theta}_k^T \phi(\mathbf{x})^{(j)} + b_k \Big) \Big), && \text{(Normalizing constant)} \\
& + \frac{C_1}{2} \sum_{k=1}^{K-1} \|\boldsymbol{\theta}_k\|_2^2 && \text{(Regularizer),}
\end{aligned}
$$

where $(\boldsymbol{\theta}_k,\ b_k)$ are trainable parameters associated with the kth timepoint, $D = \{T^{(j)}, \delta^{(j)}, \mathbf{x}^{(j)}\}_{j=1}^{N}$ is a dataset with N patients, N_c of whom are censored and y_k is the binary event indicator for timepoint k. We define $\phi(x)$ to be a multi-layer perceptron with exponential linear unit (ELU) activations[2,3].

The model takes a vector of EMR features and volume as input and predicts the probability of event occurring within each of the discrete time intervals. An individual survival curve can be computed for each patient from the predictions, from which we read out the probability of 2-year survival, as well as the lifetime risk score. Tumour volume was computed using the mesh algorithm implemented in PyRadiomics version 2.2.0. We used one-hot encoding for categorical EMR features and encoded any missing values as separate dummy category. The inputs were normalized to zero mean and unit variance using statistics computed on the training set. We implemented the MTLR algorithm using using the PyTorch framework version 1.5.0 and trained it for 100 epochs using the Adam optimizer with default momentum parameters[4]. The number of MTLR time bins was set as $\sqrt{N_{\text{uncensored}}}$, where $N_{\text{uncensored}}$ is the number of uncensored patients in the training set and the bin edges were set to quantiles of training survival time distribution. We tuned other hyperparameters by maximizing 5-fold cross validation AUROC on the training set using 60 iterations of random search. The final hyperparameter settings are shown in the table below.

Hyperparameter	Value
Batch size	512
Dropout	.24
Hidden layer sizes	(128)
C_1	10
Learning rate	.006
Weight decay	6×10^{-5}

Table S1. Submission 1 hyperparameters.

Submission 2

We applied the fuzzy prediction framework used by submission 4 to EMR variables only (no engineered radiomics except tumour volume used as the fuzzy variable).

Submission 3

We attempted to identify prognostic signal in radiomic features beyond tumour volume using a fuzzy learning approach[5]. Briefly, we divided the training dataset into 2 groups based on the tumour volume (greater/less than median) and trained a binary logistic regression model to predict the probability of falling in the large volume group from the input features. We then built a separate logistic regression (for the binary task) and Cox proportional hazard (for the lifetime survival task) within each of the subgroups. The final predictions were computed as a linear combination of the subgroup model predictions weighted by the predicted probability of falling within the subgroup. We used the provided EMR features as inputs together with hand-engineered imaging features. All categorical features were converted to indicator variables (with a separate category for missing values) and continuous features were normalized to zero mean and unit variance. To reduce the dimensionality of feature space and prevent the inclusion of potentially noisy variables, we used a curated set of radiomic features which were previously found to be prognostic in HNC: `GLSZM-SizeZoneNonUniformity`, `GLSZM-ZoneVariance`, `GLRLM-LongRunHighGrayLevelEmphasis`[6,7]. We extracted the features using PyRadiomics 2.2.0 with fixed quantization bin width equal to 25 and after resampling the images to isotropic 1mm voxel spacing. Standard logistic regression (as implemented in Scikit-learn package[8], version 0.22.1) with LBFGS solver and inverse frequency weighted loss function was used for binarized output, Cox modeling (Lifelines package, version 0.24.5) with partial hazard fitting and step size of 0.5 was used for survival prediction.

Submission 4

The same algorithm as submission 1 but with EMR data only and different network architecture.

Hyperparameter	Value
Batch size	1024
Dropout	.14
Hidden layer sizes	(32, 32, 32)
C_1	10
Learning rate	.006
Weight decay	1.3×10^{-6}

Table S1. Submission 3 hyperparameters.

Submission 5

Our approach uses a 3D convnet with EMR features concatenated before fully-connected layers. The convnet uses `conv-batch norm-leaky ReLU` block structure with negative activation slope equal to .1. The network takes a 50mm $\times$ 50mm $\times$ 50mm image patch centred on the GTV mask centroid as input and outputs the predicted probability of death before 2 years. The images were resampled to isotropic 1mm spacing, intensity clipped to [-500 HU, 1000 HU] range and normalized by subtracting the training set mean and dividing by training standard deviation. EMR features were normalized

analogously and categorical features were additionally one-hot encoded, replacing any missing values with a special 'missing' category. We applied random data augmentation, including random in-plane rotations between $(-\pi/6, \pi/6)$ radians, random flipping along the x and z axes and additive Gaussian noise with standard deviation .05 (after normalization). We implemented the model using PyTorch 1.5.0 and PyTorch Lightning 0.7.6 and trained it for 500 epochs using the Adam algorithm with batch size 10 and learning rate 10^{-3} decayed by a factor of 10 after 60, 160 and 360 epochs. To prevent overfitting, we used dropout with probability .4 and weight decay regularization with coefficient 10^{-4}. We used binary cross entropy loss weighted by the inverse frequency of positive label to reduce the impact of class imbalance. The hyperparameters were selected based on performance on a 10% validation set held out from the training set.

Operation	Output channels	Kernel size
Convolution	64	5^3
Convolution	128	3^3
Max pooling (stride=2)	-	2^3
Convolution	256	3^3
Convolution	512	3^3
Max pooling (stride=2)	-	2^3
Global average pooling	-	-
Fully-connected	512	-
Concatenate EMR features	-	-
Fully-connected	512	-
Dropout (p=.4)	-	-
Fully-connected	1	-

Table S1. Convnet architecture. Each convolution operation corresponds to the block described above.

Submission 6

We used a 2D convnet analogously to submission 10 and combined the learned image features with EMR variables. The EMR features were one-hot encoded and normalized to zero mean and unit variance before being passed through three fully-connected layers with 8 hidden units and concatenated with the convnet output before the final classification layer.

Submission 7

To learn prognostic image representations, we used a 3D dense convolutional network (DenseNet) with multitask learning prediction head. The network takes a cropped 60mm $\times$ 60mm $\times$ 60mm image patch, centred on the GTV centroid and outputs the probability of death at multiple discrete time intervals. We used convolutional block structure previously validated in retinal tomography scans[9]. Each convolutional block consists of multiple layers of within-slice ($1 \times 3 \times 3$) and across-slice ($3 \times 1 \times 1$) convolutions, followed by batch normalization[10] and ReLU nonlinearities. The EMR features were concatenated with the convnet output before the final prediction layer. The convnet architecture is shown in table S1 below.

Block	Operations
Convolution	$\begin{bmatrix}\text{conv } 1\times3\times3\\ \text{conv } 1\times3\times3\end{bmatrix}$
Pooling	max pool $2\times2\times2$, stride 2
Dense block 1	$\begin{bmatrix}\text{conv1}\times3\times3\\ \text{conv1}\times3\times3\\ \text{conv3}\times1\times1\end{bmatrix}\times2$
Transition 1	conv $1\times1\times1$ max pool $2\times2\times2$, stride 2
Dense block 2	$\begin{bmatrix}\text{conv1}\times3\times3\\ \text{conv1}\times3\times3\\ \text{conv3}\times1\times1\end{bmatrix}\times2$
Transition 2	conv $1\times1\times1$ max pool $2\times2\times2$, stride 2
Dense block 3	$\begin{bmatrix}\text{conv1}\times3\times3\\ \text{conv1}\times3\times3\\ \text{conv3}\times1\times1\end{bmatrix}\times2$
Transition 3	conv $1\times1\times1$ max pool $2\times2\times2$, stride 2
Dense block 4	$\begin{bmatrix}\text{conv1}\times3\times3\\ \text{conv1}\times3\times3\\ \text{conv3}\times1\times1\end{bmatrix}\times3$
Global pool	adaptive average pool
EMR features	concat(input, EMR features)
Output	MTLR(time bins=40)

Table S1. 3D Dense Net architecture. The convolution kernel sizes are given as (depth, width, height). Each conv operation above corresponds to the sequence batch norm-ReLU-conv, except for the first 2 convolutions where the order is reversed. Additionally, we applied dropout after each transition layer. Note that the number of channels in each layer is determined by the first convolution output channels (here 32) and the growth rate (tunable hyperparameter).

The input image patches were resampled to 3mm × 1mm × 1mm voxel spacing and clipped to range [-500, 1000] HU. For EMR features, we one-hot encoded categorical inputs features, with any missing values represented as separate category. Both images and EMR features were normalized to zero mean and unit variance using statistics computed on the training set. 10% of training patients were set aside as a validation set for hyperparameter tuning. We applied random data augmentation to input image patches (table S1). The augmentation operations were implemented in SimpleITK version 1.2.4 and fused into a single transform to minimize interpolation artifacts.

Parameter	Value or range
In-plane rotation	[-10°–10°]
Flip along z-axis	[True, False]
In-plane shear	[-.005, .005]
In-plane scaling	[.8, 1.2]
In-plane translation	[-10mm, 10mm]
In-plane elastic deformation	grid size = $(2, 2)$, $\alpha = 5$
Gaussian noise	$\mu = 0$, $\sigma = 10$

Table S1. Data augmentation operations used during training. Parameter values were drawn randomly for each input from the ranges shown above.

We implemented our approach using PyTorch version 1.5.0 and PyTorch Lightning version 0.7.6.

The model was trained by minimizing the MTLR negative log likelihood using the Adam optimizer with default momentum parameters for a maximum of 200 epochs. Training was stopped early if the loss on the validation set did not improve by at least .0005 for 20 epochs. The initial learning rate was decayed by a factor of .5 after 60, 100, 140 and 180 epochs. To mitigate the high class imbalance present in the dataset, we oversampled the minority class by using a balanced minibatch sampler, which helped to stabilize training. Hyperparameters were selected by maximizing the validation set performance using 60 iterations of random search. The final hyperparameter configuration is shown below.

Hyperparameter	Value
Batch size	8
Dense block layers	(2, 2, 2, 3)
DenseNet growth rate	24
Dropout	.38
Initial num. channels	32
MTLR regularization	10
Learning rate	.0002
Weight decay	9.4×10^{-4}

Table S1. Submission 7 hyperparameters.

Submission 8

We trained a three-layer neural network with scaled exponential linear unit (SELU) activation and alpha-dropout[11]. The inputs were EMR features after one-hot encoding (for categorical features) and normalization (for continuous features) and the output was the predicted 2-year survival probability. To address the issue of class imbalance, we used biased sampling to adjust the frequency of positive training samples. We selected the hyperparameters manually based on cross-validation performance.

Submission 9

We used a 3D dense convolutional network (DenseNet) with similar block structure and architecture as submission 7. The main differences were the lack of EMR inputs and a second context network, taking a downsampled image patch with the same location as the base network but 2× lower resolution, providing a zoomed-out view of the tumour surroundings. The feature maps from both streams were concatenated along the channel dimension before global pooling and passed through additional $1 \times 1 \times 1$ convolution to maintain equal number of channels. The final hyperparameter configuration is shown below.

Hyperparameter	Value
Batch size	16
Dense block layers	(2, 2, 2, 3)
DenseNet growth rate	24
Dropout	.03
Initial num. channels	32
MTLR regularization	10
Learning rate	.00027
Weight decay	1.7×10^{-4}

Table S1. Submission 9 hyperparameters.

Submission 10

The architecture used was a VGGNet[12] with batch normalization and sigmoid output for binary classification. The inputs were formed by extracting the largest 2D GTV slice and concatenating with the binary mask along the channel axis. The images were cropped to 96×96 pixel window centred on the mask centroid, intensity clipped to [-1000, 400] HU range and normalized to zero mean and unit variance. We applied data augmentation including additive Gaussian noise (image channel only) with standard deviation of 12 HU, random translations between ± 20 pixels in each direction and random

scaling between .85 and 1.25. The model was implemented in PyTorch and trained with minority class oversampling to mitigate class imbalance.

Submission 11

We used similar training setup and implementation as submission 5 but relying on images only (without EMR features) and a different convnet architecture (see below).

Operation	Output channels	Kernel size
Convolution	64	3^3
Convolution	128	3^3
Convolution	128	3^3
Max pooling (stride=2)	-	2^3
Convolution	256	3^3
Convolution	256	3^3
Convolution	512	3^3
Max pooling (stride=2)	-	2^3
Convolution	512	3^3
Convolution	1024	3^3
Convolution	1024	3^3
Max pooling (stride=2)	-	2^3
Global average pooling	-	-
Fully-connected	1024	-
Dropout (p=.4)	-	-
Fully-connected	1	-

Table S1. Convnet architecture.

Submission 12

We applied the fuzzy training framework used in submissions 2 and 3 to the curated set of radiomic features only.

Additional results

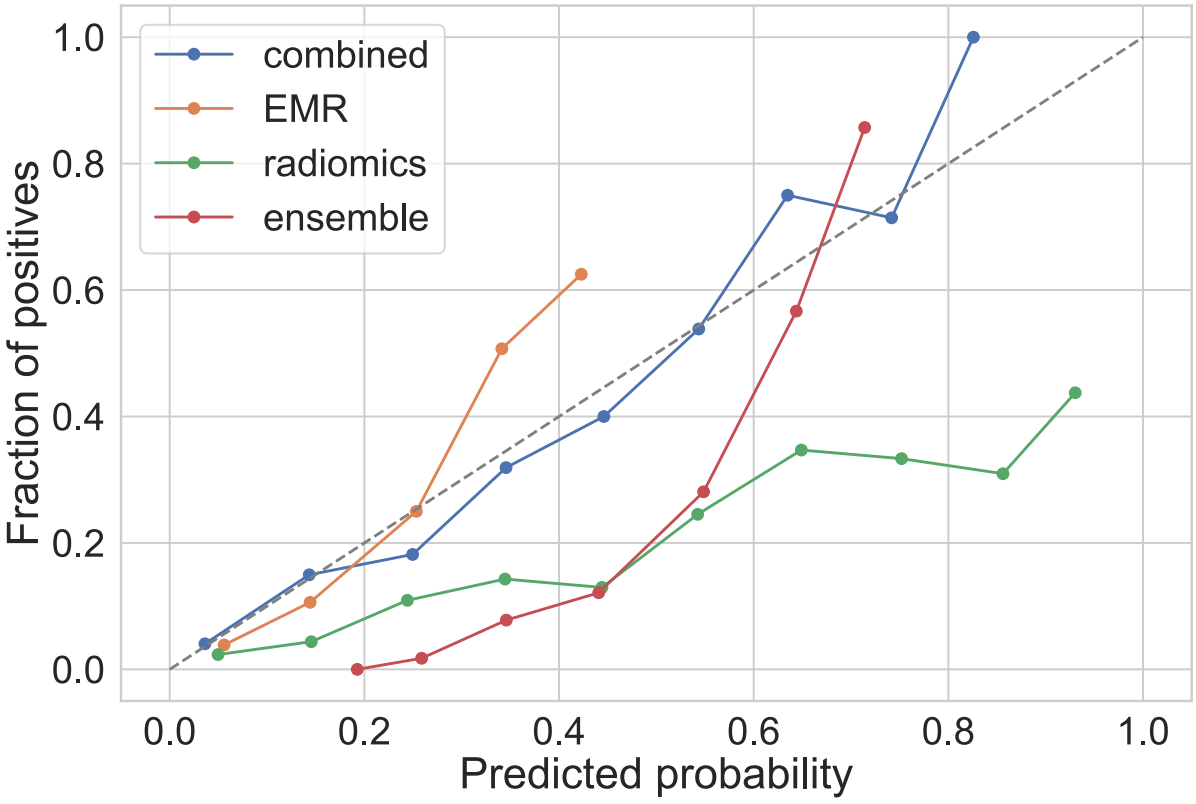

Figure S2. Calibration of predicted 2-year event probabilities for the best performing model in each category and the ensemble of all models.

References

1. Yu, C.-N., Greiner, R., Lin, H.-C. & Baracos, V. in *Advances in Neural Information Processing Systems 24* (eds Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F. & Weinberger, K. Q.) 1845–1853 (Curran Associates, Inc., 2011). http://papers.nips.cc/paper/4210-learning-patient-specific-cancer-survival-distributions-as-a-sequence-of-dependent-regressors.pdf.
2. Fotso, S. *Deep Neural Networks for Survival Analysis Based on a Multi-Task Framework* arXiv: 1801.05512 [cs, stat]. http://arxiv.org/abs/1801.05512 (2020).
3. Clevert, D.-A., Unterthiner, T. & Hochreiter, S. *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)* arXiv: 1511.07289 [cs]. http://arxiv.org/abs/1511.07289 (2020).
4. Kingma, D. P. & Ba, J. *Adam: A Method for Stochastic Optimization* arXiv: 1412.6980 [cs]. http://arxiv.org/abs/1412.6980 (2019).
5. Haibe-Kains, B. *et al.* A Fuzzy Gene Expression-Based Computational Approach Improves Breast Cancer Prognostication. *Genome Biol* **11,** R18. ISSN: 1465-6906. http://genomebiology.biomedcentral.com/articles/10.1186/gb-2010-11-2-r18 (2021) (2010).
6. Vallières, M. *et al.* Radiomics Strategies for Risk Assessment of Tumour Failure in Head-and-Neck Cancer. *Sci Rep* **7,** 10117. ISSN: 2045-2322. http://www.nature.com/articles/s41598-017-10371-5 (2019) (Dec. 2017).
7. Diamant, A., Chatterjee, A., Vallières, M., Shenouda, G. & Seuntjens, J. Deep Learning in Head & Neck Cancer Outcome Prediction. *Scientific Reports* **9.** ISSN: 2045-2322. http://www.nature.com/articles/s41598-019-39206-1 (2019) (Dec. 2019).
8. Pedregosa, F. *et al.* Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011).

9. De Fauw, J. *et al.* Clinically Applicable Deep Learning for Diagnosis and Referral in Retinal Disease. *Nat Med* **24,** 1342–1350. ISSN: 1078-8956, 1546-170X. http://www.nature.com/articles/s41591-018-0107-6 (2019) (Sept. 2018).

10. Ioffe, S. & Szegedy, C. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift* arXiv: 1502.03167 [cs]. http://arxiv.org/abs/1502.03167 (2019).

11. Klambauer, G., Unterthiner, T., Mayr, A. & Hochreiter, S. *Self-Normalizing Neural Networks* arXiv: 1706.02515 [cs, stat]. http://arxiv.org/abs/1706.02515 (2019).

12. Simonyan, K. & Zisserman, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition* arXiv: 1409.1556 [cs]. http://arxiv.org/abs/1409.1556 (2021).