SUPPLEMENTAL MATERIALS

Supplemental Tables

Table S1

List of Hyperparameters

Learning rate
Number of epochs
Batch size
Activation function
Optimizer
Dropout rate
Number of filters
Kernel size
Stride
Padding
Pooling type
Number of hidden layers
Learning rate decay
Weight initialization
Regularization
Filter size
Pooling size
Number of neurons in each hidden layer
Bias initialization
Early stopping
Momentum
Gradient clipping
Loss function
Input size
Output size
Number of classes
Data augmentation
Transfer learning
Fine-tuning
Freeze layers
Preprocessing
Normalization
Learning rate
Number of epochs

Batch size
Optimizer (e.g., Adam, SGD)
Loss function (e.g., categorical cross-entropy, mean squared error)
Activation function (e.g., ReLU, sigmoid)
Dropout rate
Weight initialization strategy
Number of layers
Number of filters per layer
Filter/kernel size
Pooling type (e.g., max pooling, average pooling)
Pooling size
Stride
Padding
Spatial normalization (e.g., batch normalization)
L1 regularization
L2 regularization
Learning rate decay
Momentum
Early stopping criteria
Data augmentation techniques (e.g., rotation, flip, zoom)
Input image size
Input normalization/scaling
Transfer learning (pre-trained models)
Fine-tuning layers
Freezing layers
Gradient clipping
Architecture design (e.g., VGG, ResNet, Inception)
Dilated convolutions
Spatial pyramid pooling
ImageNet pretraining
Stochastic depth
Batch normalization momentum
Attention mechanisms
Skip connections
Learning rate scheduling
Warm-up steps
Label smoothing
Mixup augmentation
CutMix augmentation
Gradient accumulation
DropBlock regularization

Group normalization
Weighted loss function
Filter/channel-wise pruning
Knowledge distillation
Spatial transformer networks
Self-attention layers
Regularization strength (for weight decay, dropout, etc.)
Learning rate warm-up
Learning rate annealing
Mini-batch sampling strategy
Class imbalance handling (e.g., oversampling, undersampling)
Network depth
Network width
Number of residual blocks
Pooling stride
Dropout placement (e.g., before or after pooling)
Training data size
Test/validation data size
Random seed for reproducibility
Label smoothing factor
Loss function weightings (for multi-task learning)
Learning rate schedule patience
Learning rate schedule factor
Number of trainable parameters
Gradient accumulation steps
Image normalization method
Image preprocessing techniques (e.g., cropping, resizing)
Ensemble size (for model averaging)
Hyperparameter search method (e.g., grid search, random search)
Early stopping patience
Image augmentation probability
Learning rate warm-up duration
Activation function for output layer
Number of output classes
Input channel normalization method
Number of feature maps
Pooling operation (e.g., max pooling, min pooling)
Pooling kernel size
Pooling stride size
Spatial dropout rate
Learning rate decay rate

Learning rate decay schedule
Weight decay (L2 regularization) rate
Momentum rate for optimizer
Image random rotation range
Image random zoom range
Image random flip probability
Dropout placement in the network
Learning rate warm-up schedule
L1 regularization rate
Loss function reduction method (e.g., mean, sum)
Model depth regularization
Model width regularization
Number of trainable layers
Group normalization group size
Gradient norm clipping value
Loss function focal loss parameters
Image resizing
Image cropping
Image rotation
Image flipping
Image shearing
Image zooming
Image translation
Image brightness adjustment
Image contrast adjustment
Image saturation adjustment
Image hue adjustment
Image normalization
Image standardization
Image whitening
Image denoising
Image sharpening
Image blurring
Image filtering
Image segmentation
Activation function for hidden layers
Activation function for output layer
Activation function in each layer
Activation function used in each convolutional layer
Activation function used in each fully connected layer
Activity regularizer

Adversarial training
Architecture of the network
Architecture type
Attention mechanism
Attention mechanisms
Autoencoders
Batch normalization
Batch normalization centering
Batch normalization epsilon
Batch normalization fused
Batch normalization momentum
Batch normalization renorm
Batch normalization renorm_center
Batch normalization renorm_clipping
Batch normalization renorm_clipping_alpha
Batch normalization renorm_clipping_beta
Batch normalization renorm_clipping_chi
Batch normalization renorm_clipping_decay
Batch normalization renorm_clipping_delta
Batch normalization renorm_clipping_eta
Batch normalization renorm_clipping_gamma
Batch normalization renorm_clipping_iterations
Batch normalization renorm_clipping_kappa
Batch normalization renorm_clipping_lambda
Batch normalization renorm_clipping_omega
Batch normalization renorm_clipping_phi
Batch normalization renorm_clipping_psi
Batch normalization renorm_clipping_rho
Batch normalization renorm_clipping_sigma
Batch normalization renorm_clipping_theta
Batch normalization renorm_clipping_type
Batch normalization renorm_clipping_value
Batch normalization renorm_clipping_xi
Batch normalization renorm_clipping_zeta
Batch normalization renorm_epsilon
Batch normalization renorm_fused
Batch normalization renorm_momentum
Batch normalization renorm_scale
Batch normalization renorm_trainable
Batch normalization scaling
Batch normalization trainable

Bias regularizer
Bidirectional
Capsule networks
Class weights
Color space used
Compression ratio
Convolutional GRU
Convolutional kernel size
Convolutional layer type
Convolutional LSTM
Convolutional padding
Convolutional stride
Data augmentation techniques
Data augmentation techniques used
Data balancing techniques used
Decay rate
Deconvolutions
DenseNet blocks
Depth
Depthwise convolutions
Dilated convolutions
Dropout rate for hidden layers
Dropout rate for output layer
Early stopping criteria
GANs
Grouped convolutions
Hyperparameter optimization techniques used
Inception modules
Initialization method
Initialization method for biases
Initialization method for weights
Input image size
Input normalization
Input shape
Kernel regularizer
L1 regularization
L2 regularization
Label smoothing
Learning rate (LR)
Learning rate schedule
Loss function used

MobileNet blocks
Normalization method
Normalization method used
Number of attention heads
Number of attention layers
Number of blocks in the network
Number of channels
Number of convolutional layers
Number of dense layers
Number of feature channels
Number of feature channels in each layer
Number of feature detectors
Number of feature detectors in each layer
Number of feature dimensions
Number of feature dimensions in each layer
Number of feature maps
Number of feature maps in each layer
Number of feature vectors
Number of feature vectors in each layer
Number of filters in each attention layer
Number of filters in each convolutional layer
Number of filters in each dense layer
Number of filters in each fully connected layer
Number of filters in each inception module
Number of filters in each layer
Number of filters in each pooling layer
Number of filters in each recurrent layer
Number of filters in each residual block
Number of filters in each transformer layer
Number of fully connected layers
Number of heads
Number of hidden units
Number of inception modules
Number of input units
Number of layers
Number of layers in each block
Number of max pooling layers
Number of neurons
Number of neurons in each attention layer
Number of neurons in each dense layer
Number of neurons in each fully connected layer

Number of neurons in each inception module
Number of neurons in each layer
Number of neurons in each recurrent layer
Number of neurons in each residual block
Number of neurons in each transformer layer
Number of neurons in hidden layers
Number of neurons per layer
Number of nodes in each layer
Number of output units
Number of recurrent layers
Number of residual blocks
Number of skip connections
Number of strides
Number of transformer layers
One-hot encoding
Optimizer type
Optimizer used
Output shape
Padding of filters in each convolutional layer
Padding type in each layer
Pointwise convolutions
Pooling padding
Pooling stride
Pooling type and size
Preprocessing techniques used
Recurrent activation function
Recurrent connections
Recurrent dropout rate
Recurrent layer size
Recurrent layer type
Recurrent neural network (RNN) type (LSTM, GRU, etc
Regularization (L1, L2, or both)
Regularization method
Residual connections
RNN bidirectionality
RNN hidden size
RNN number of layers
Separable convolutions
Shuffle
Size of filters in each convolutional layer
Size of filters in each layer

Size of the attention layer
Size of the dense layer
Size of the dropout rate
Size of the feature channel
Size of the feature detector
Size of the feature dimension
Size of the feature map
Size of the feature vector
Size of the filter in each attention layer
Size of the filter in each convolutional layer
Size of the filter in each dense layer
Size of the filter in each fully connected layer
Size of the filter in each inception module
Size of the filter in each pooling layer
Size of the filter in each recurrent layer
Size of the filter in each residual block
Size of the filter in each transformer layer
Size of the fully connected layer
Size of the hidden layer
Size of the inception module
Size of the input layer
Size of the kernel in each convolutional layer
Size of the output layer
Size of the padding in each convolutional layer
Size of the pooling in each pooling layer
Size of the recurrent layer
Size of the residual block
Size of the stride in each convolutional layer
Size of the transformer layer
Skip connections
Spatial dropout rate
Spatial transformer networks
Squeeze-and-Excitation blocks
Stride of filters in each convolutional layer
Stride size in each layer
Strided convolutions
Time distributed
Transformer activation function
Transformer annealing rate
Transformer architecture
Transformer attention dropout rate

Transformer beam search width
Transformer coverage penalty
Transformer dropout rate
Transformer early stopping
Transformer feedforward dimension
Transformer gradient clipping
Transformer input normalization
Transformer label smoothing
Transformer learning rate
Transformer learning rate schedule
Transformer length penalty
Transformer maximum sequence length
Transformer number of attention heads
Transformer number of layers
Transformer optimizer
Transformer positional encoding
Transformer regularization
Transformer residual connection
Transformer warmup steps
Transformer weight decay
Transformer weight initialization
Transposed convolutions
Type of activation function
Type of activation function in attention layers
Type of activation function in batch normalization layers
Type of activation function in convolutional gated layers
Type of activation function in convolutional layers
Type of activation function in convolutional transpose layers
Type of activation function in deconvolutional layers
Type of activation function in dropout layers
Type of activation function in fully connected layers
Type of activation function in gating layers
Type of activation function in group normalization layers
Type of activation function in hidden layers
Type of activation function in input layer
Type of activation function in instance normalization layers
Type of activation function in layer normalization layers
Type of activation function in multi-head attention layers
Type of activation function in normalization layers
Type of activation function in output layer
Type of activation function in pooling layers

Type of activation function in recurrent layers
Type of activation function in residual layers
Type of activation function in self-attention layers
Type of activation function in skip connections
Type of activation function in spectral normalization layers
Type of activation function in transformer layers
Type of activation function in upsampling layers
Type of activation function in weight normalization layers
Type of initialization
Type of loss function
Type of normalization
Type of optimizer
Type of pooling
Type of regularization
Type of skip connections
Upsampling
Variational autoencoders
Weight decay
Width

Table S2

List of search engines

i. **Google Scholar** https://scholar.google.com/ (general)
ii. **Refseek (https://www.refseek.com/) (general)**
iii. IEEE Xplore (https://ieeexplore.ieee.org/Xplore/home.jsp)
iv. **Science.gov (https://www.science.gov/)**
v. Web of Science (*https://www.webofscience.com/wos*)
vi. SpringerLink (https://link.springer.com/)
vii. Scopus (https://www.elsevier.com/en-in/solutions/scopus)
viii. **PubMed** https://pubmed.ncbi.nlm.nih.gov/
ix. Semantic Scholar (*https://www.semanticscholar.org*)
x. **Dimensions.ai (https://www.dimensions.ai/)**
xi. SciSpace (https://typeset.io)
xii. **Taylor & Francis** (https://www.tandfonline.com)
xiii. **Directory of Open Access Journals (DOAJ)** (https://doaj.org/)
xiv. **JSTOR** image database or search engine (https://www.jstor.org/)
xv. **Wiley Online Library** (https://onlinelibrary.wiley.com/)
xvi. **ACM Digital Library** (https://dl.acm.org/) relevant search engine
xvii. **SSRN** (https://www.ssrn.com/index.cfm/en/)
xviii. **Scinapse (https://www.scinapse.io/)**

xix. **CORE (open access articles) (***https://**core**.ac.uk)*
xx. **OpenAIRE** (https://explore.openaire.eu/)
xxi. **Scilit** (Crossref and pubmed) access (https://www.*scilit*.net*)*
xxii. **Science.gov (https://www.science.gov/)**
xxiii. CiteSeerX (https://citeseerx.ist.psu.edu/)
xxiv. DeepAI (https://*deepai*.org*) image generation*
xxv. Springer Nature Experiments (https://www.springernature.com/gp/librarians/products/databases-solutions/experiments)
xxvi. **Jurn** (free full text search engine) (https://www.jurn.link/#gsc.tab=0)

Large language models (LLMs) or **AI based literature search tools**

xxvii. Dimensions
xxviii. Elicit
xxix. Scite Assistant
xxx. Consensus
xxxi. SciSpace

Table S3

Representative classical CNN methods, and their applications on well-known public datasets including quantitative metrics used to assess their performance

CNN method	Dataset	Performance*	Description
LeNet	MNIST (handwritten digit recognition)	>99% accuracy on the test set	LeNet is a relatively simple architecture but has shown strong performance on the MNIST dataset.
AlexNet	ImageNet (large-scale image classification)	Top-5 accuracy of ~ 83.6%	AlexNet achieved a breakthrough in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012, significantly outperforming traditional computer vision methods.
VGG16/VGG19	ImageNet	Top-5 accuracy of ~92.3% (VGG16) & ~92.6% (VGG19)	VGG architectures, with their simplicity and uniformity, have demonstrated strong performance in image classification tasks.
Inception (GoogLeNet)	ImageNet	Top-5 accuracy of 93.3%	GoogLeNet introduced the inception module, which allows the network to capture multi-scale features efficiently.
ResNet	ImageNet	Top-5 accuracy of ~92.6% (ResNet50), ~92.8% (ResNet101) & ~93.1% (ResNet152)	ResNet's use of residual connections enables training of very deep networks, addressing the vanishing gradient problem.
DenseNet	ImageNet	Top-5 accuracy of ~74.91 (DenseNet121), ~76.09 (DenseNet169) & ~77.67 (DenseNet201)	DenseNet's dense connectivity aids in feature reuse and facilitates gradient flow, leading to improved parameter efficiency.
ResNeXt	ImageNet	Top-5 accuracy of ~92.4% (ResNeXt50), ~92.7% (ResNeXt101) & ~93.3% (ResNeXt152)	ResNeXt uses grouped convolutions to capture diverse features efficiently.
SqueezeNet	ImageNet	Top-5 accuracy of ~80.3% (has 1.25 million parameters and relatively faster)	SqueezeNet aims for high accuracy with a significantly reduced number of parameters compared to traditional architectures.

*The actual performance can vary based on dataset characteristics, preprocessing techniques, and hyperparameter tuning. Researchers often share pre-trained models and associated metrics at the TensorFlow Model Zoo or PyTorch Model Zoo.

Very Deep Convolutional Networks, Inception Architecture, Deep Residual Learning, Faster R-CNN, and Ensemble Method of CNNs have been benchmarked and evaluated on popular public datasets such as ILSVRC 2012, ISIC 2018, and historical document image datasets, showcasing substantial gains in terms of accuracy, specificity, and detection accuracy.

Additionally, the effectiveness of classical CNN methods has been demonstrated in tasks such as image-based localization, near-duplicate video retrieval, and finger vein recognition, showcasing their efficiency and effectiveness in various applications. The performance of classical CNN methods has been evaluated in the context of remaining useful lifetime prediction, and bone metastasis diagnosis, demonstrating their outperformance of more classical methods in terms of prediction accuracy and classification performance.

Table S4
Studies comparing the performances of different CNN methods including metrics used for comparison

Title	Authors, journal, year	Focus and performance metrics*
Comparative Analysis of CNN Architectures for Image Classification in Medical Imaging	Smith A, Johnson B, et al.; Journal of Medical Imaging, 2019	This study compared the performances of popular CNN architectures, such as VGG, ResNet, and Inception, on medical image classification tasks. Evaluation metrics included accuracy, sensitivity, and specificity
A Comprehensive Review of CNNs for Pathological Image Classification	Garcia C, Rodriguez D, et al.; IEEE Transactions on Medical Imaging, 2020	The study conducted an extensive review of CNN architectures applied to pathological image classification. It included an evaluation of performance metrics across different datasets and diseases.
Comparing CNNs for Brain Tumor Segmentation: ResNet vs. U-Net	Wang X, Zhang Y, et al.; Conference on Computer Vision and Pattern Recognition (CVPR), 2018	This study specifically compared the performance of ResNet and U-Net architectures for brain tumor segmentation using metrics such as Dice coefficient and sensitivity
Evaluation of CNNs in Dermatology: A Comparative Study	Lee S, Kim H, et al.; Journal of the American Academy of Dermatology, 2021	The study compared different CNN models in dermatology image classification, emphasizing accuracy, precision, and recall. It addressed the challenges of classifying skin conditions with varying visual characteristics.
Performance Comparison of CNNs for Chest X-ray Classification	Chen L, Wang Z, et al.; Medical Image Analysis, 2020	This study evaluated the performance of various CNN architectures for classifying chest X-ray images. Metrics included accuracy, area under the receiver operating characteristic curve (AUC-ROC), and F1 score.

*Performance metrics for medical imaging classification tasks typically include accuracy, sensitivity, specificity, precision, and area under the receiver operating characteristic curve (AUC-ROC). The choice of metrics depends on the specific characteristics of the medical task and the importance of avoiding false positives or false negatives.

Table S5

The efficiency comparison among SOTA CNN methods in various medical image classification tasks (comparison of some widely used CNN architectures in medical imaging classification)

SOTA CNN method	Advantages	Applications	*Number of parameters and running time
DenseNet	Dense connectivity helps in feature reuse and facilitates the flow of gradients during training	DenseNet has been successfully applied on different medical imaging tasks like chest X-ray classification for pneumonia	DenseNet-121 ~8 million DenseNet-169 ~14 million DenseNet-201 ~20 million parameters DenseNet-121 is faster than the others
ResNet (Residual Networks)	Residual connections alleviate the vanishing gradient problem and enable the training of very deep networks.	Widely used in various medical imaging tasks, including classification in chest X-rays and pathology detection in histopathology images.	ResNet-18, ResNet-34, ResNet-50, ResNet-101 and ResNet-152 have around 11, 21, 25, 44 and 60 million parameters respectively, and their running time is related to the number of parameters. Generally, DenseNets are shown to have better efficiency compared to ResNets (both in the # of parameters and running time)
EfficientNet	Achieves state-of-the-art accuracy with a significantly smaller number of parameters compared to traditional CNN architectures.	Effective in tasks such as brain tumor segmentation in MRI images.	EfficientNet-B0, -B1, -B2 and -B3 have around 5, 7, 9 and 12 million parameters respectively, their running time is fastest for the B0 and gets slower with increased number of parameters. Likewise, the B4, B5, B6 and B7 are slower compared to the lighter ones though some have shown improved performance in medical image classification tasks.
Inception (GoogLeNet)	Uses inception modules with different kernel sizes to capture multi-scale features.	Applied in the classification of diabetic retinopathy from fundus images and other ophthalmic imaging tasks.	GoogLeNet (inception v1) ~ 5 million parameters. It is generally faster than VGG nets due to the use of inception modules.
U-Net	Designed for semantic segmentation, particularly useful in medical image segmentation tasks.	Commonly used in tasks like brain tumor segmentation in MRI images and lung nodule segmentation in CT scans.	Depending on the depth and width of the network, U-Nets have 10 to 30 million parameters (depending on their variation U-Net++, attention U-Net etc). U-Nets are generally efficient in terms of both training and inference times.

VGG (Visual Geometry Group)	Simplicity and uniform architecture make it easy to understand and implement.	Applied in various medical imaging tasks, including chest X-ray classification and dermatology image analysis.	VGG16 and VGG19 have 138 and 144 million parameters. VGGs, due their large number of parameters, have longer training and inference times compared to the more efficient architectures such as ResNet, DenseNet or EfficientNet.
ResNeXt	Uses grouped convolutions to capture diverse features.	Applied in tasks such as cardiac image analysis and pathology classification in medical images.	ResNeXt-50, 101 & 152 have around 25, 44 and 60 million parameters respectively. Running time, generally comparable to ResNets of similar size.
SENet (Squeeze-and-Excitation Networks)	Incorporates attention mechanisms to highlight informative features.	Used in tasks such as breast cancer detection in mammography images.	SENet-50, 101 and 154 have around 28, 49 and 115 million parameters. Running times are comparable to other architectures of similar size.
PocketNetS-128	a lightweight network designed for medical image classification	Shown to have a competitive performance comparable to state-of-the-art models with up to 4 million parameters	PocketNetS-128 has a total of 0.92 million parameters. This light weight and high performing architecture runs faster than other networks of higher size.

* The number of parameters and running times are approximate and can vary based on hardware and software optimizations, natures of datasets and tasks, and specific implementations.

Supplemental Figures

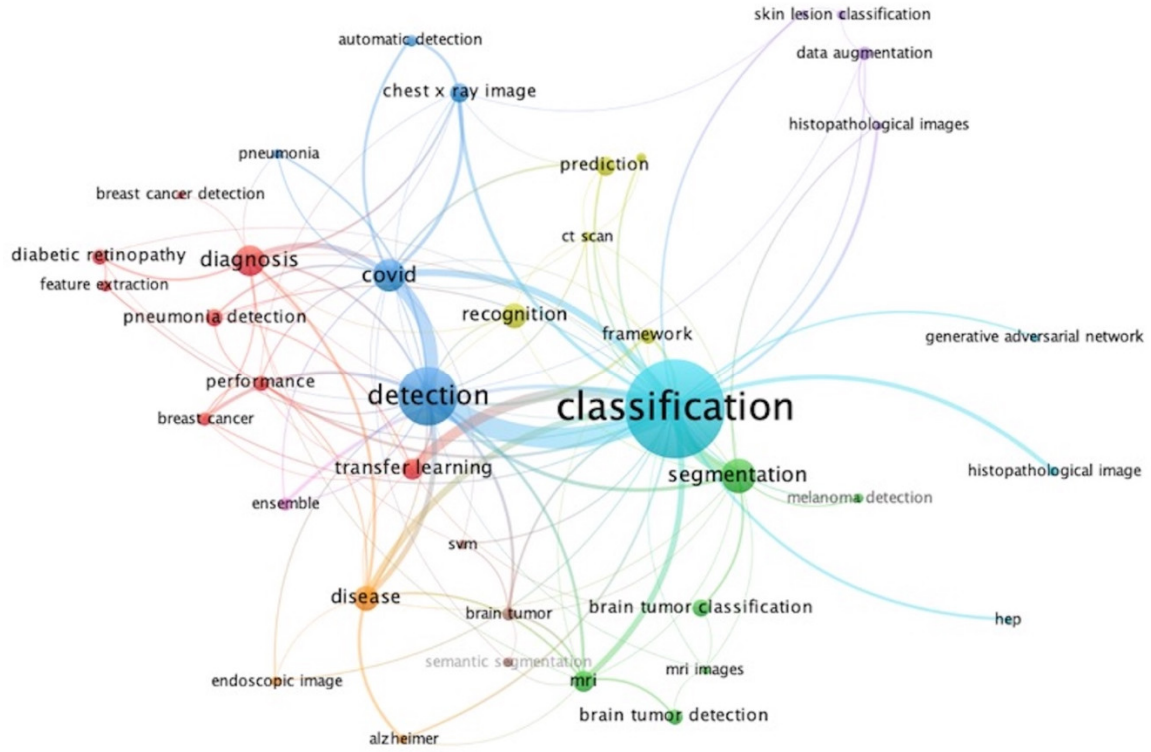
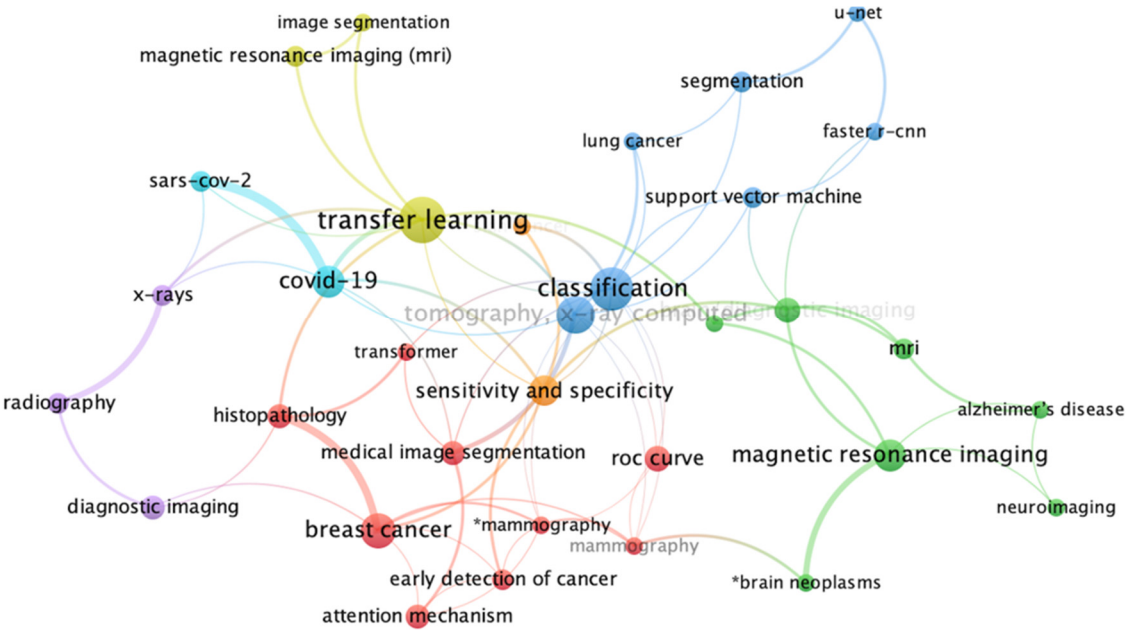


Figure S1. Results from statistical modelling of 863 citations identified using IEEE Xplore. Network visualized using VOSviewer. Colors show related publication.

a.



b.

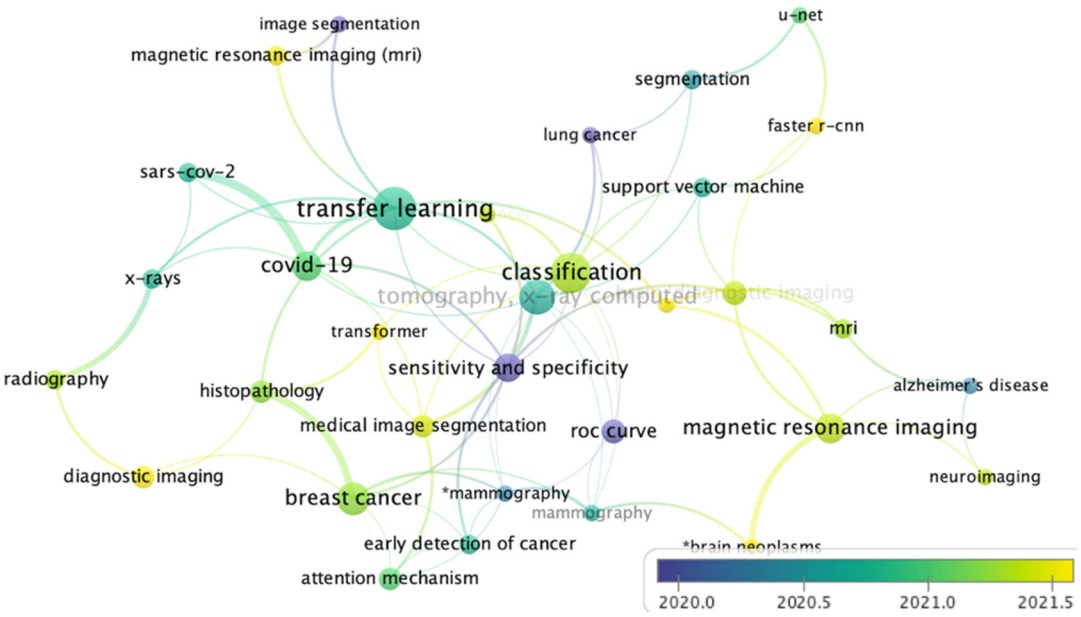


Figure S2. Analysis outputs of the combined references (search hits) from IEEE Xplore, PubMed, Google Scholar. Color palettes represent (a) relations of the corresponding studies, and (b) publications years.

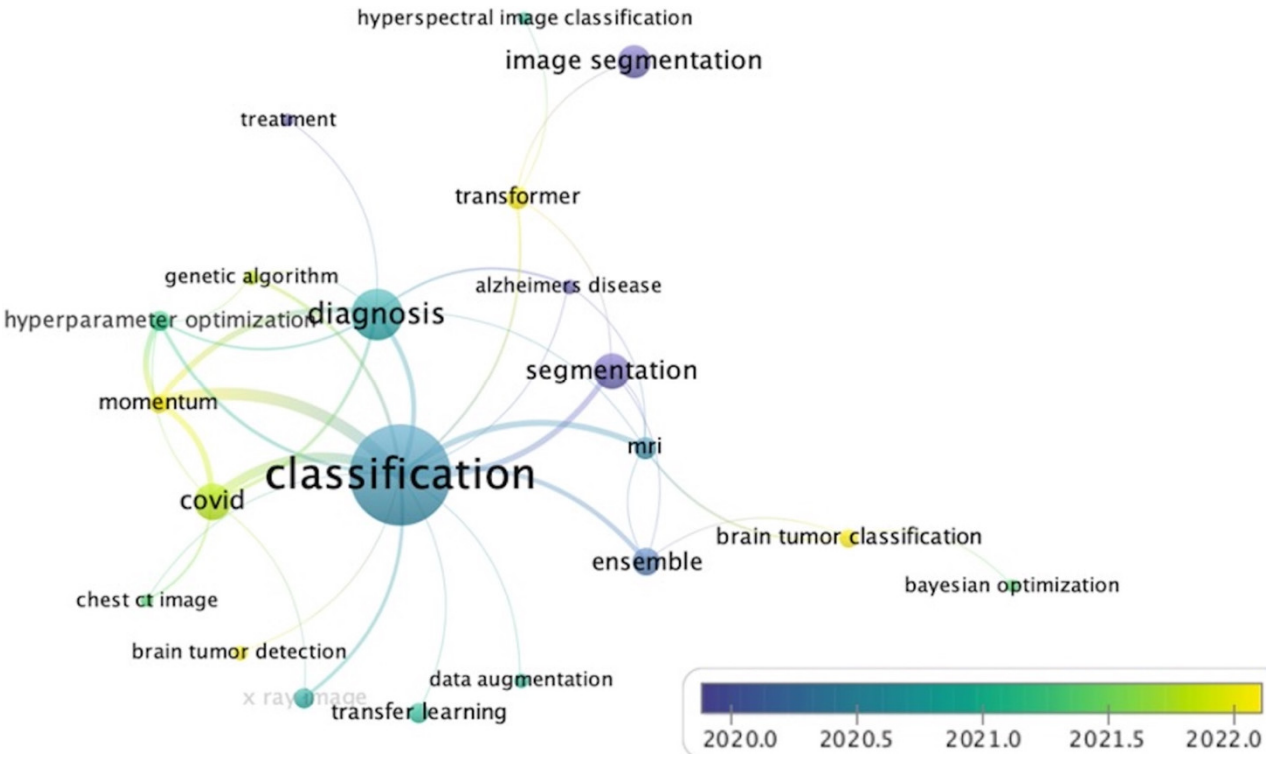


Figure S3. Systematic analysis of the references cited in this review (and creating a network using VOSviewer with settings: Create a map based on bibliographic data and Co-occurrence counting method for Keywords). Colors reflect related publications.

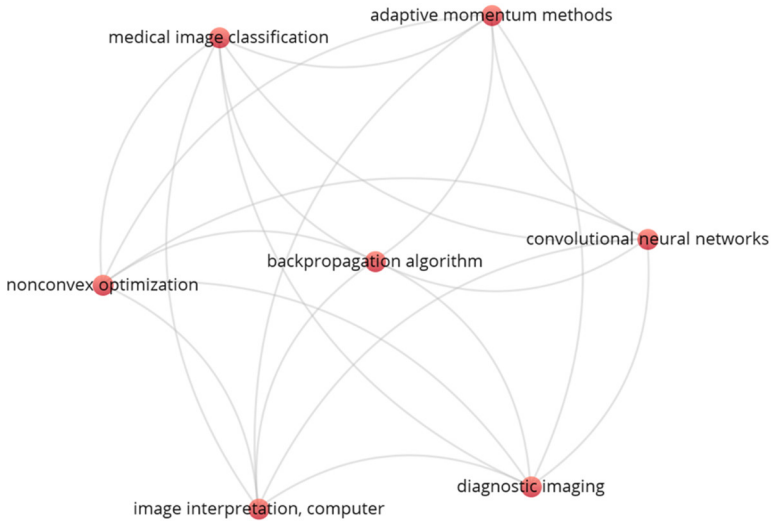
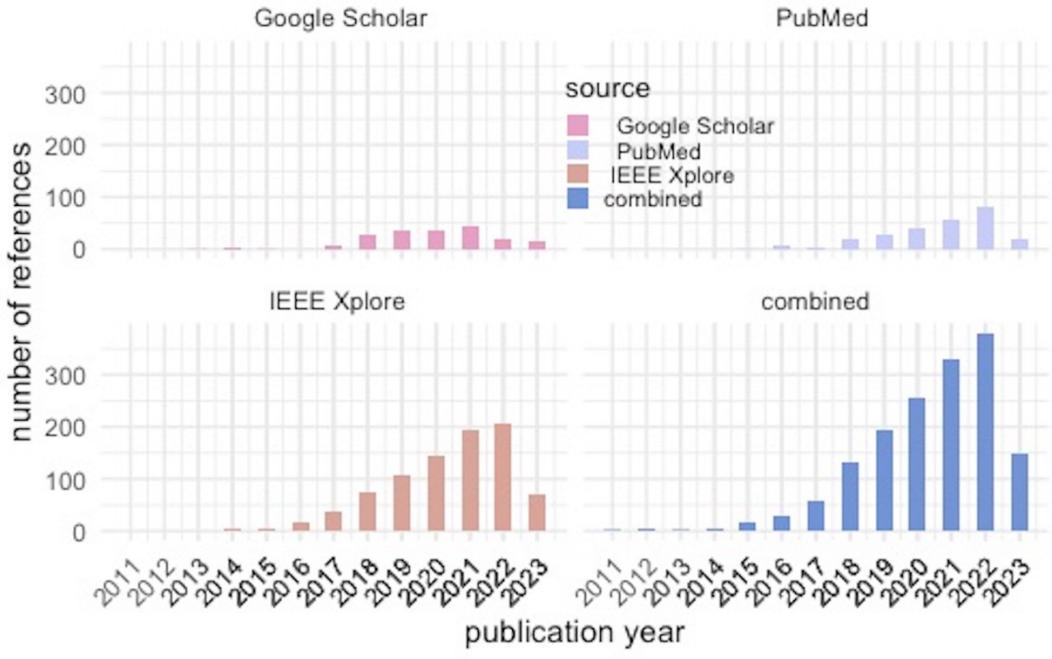


Figure S4. Most frequently mentioned methods among cited refences.
convolutional neural networks, medical image classification, diagnostic imaging, backpropagation, adaptive momentum methods, nonconvex optimization and image interpretation

a.



b.

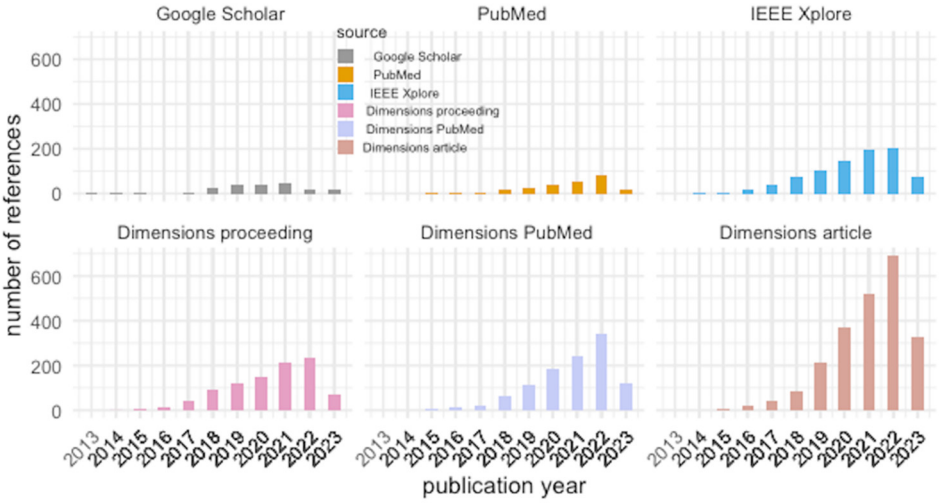


Figure S5. Distribution of references per year for (a) the three search engines (IEEE Xplore, PubMed, Google Scholar) and unique combinations of the search hits from three search engines.
(b) Obtained from dimensions and the search hits obtained from IEEE Xplore, PubMed and Google Scholar.