Classificando imagens médicas de microscopia usando redes neurais convolucionais

1st Ronald Augusto Domingos Silva
SIN 393 - Visão computacional
Universidade Federal de Viçosa
Rio Paranaíba , Brasil
Ronald.domingos@ufv.br

2nd Thalisson Lopes Gonçalves Pires
SIN 393 - Visão computacional
Universidade Federal de Viçosa
Rio Paranaíba , Brasil
thalisson.pires@ufv.br

***Abstract*—Computer vision has emerged as a field of significant impact in various practical applications, particularly in healthcare, where automated analysis of medical images can assist in disease diagnosis. This study addresses the classification problem of medical microscopy images by employing deep neural networks to differentiate between benign and malignant tissues in the lung and colon. The dataset comprises 25,000 images, and state-of-the-art deep learning models were selected for the classification task. The objective is to evaluate the effectiveness of these models in recognizing complex patterns in medical images. The findings aim to demonstrate the potential of computer vision to support medical diagnosis, highlighting the practical application of deep neural networks in real-world scenarios.**

***Index Terms*—Computer vision, deep learning, medical imaging, microscopy image classification, benign and malignant tissue, lung, colon, disease diagnosis**

Introdução

A visão computacional tem se destacado como uma área de grande impacto em diversas aplicações práticas, especialmente no campo da saúde, onde a análise automatizada de imagens médicas pode auxiliar no diagnóstico de doenças. Este trabalho aborda o problema de classificação de imagens médicas de microscopia, utilizando redes neurais profundas para diferenciar tecidos benignos e malignos nos órgãos pulmão e cólon. O dataset utilizado contém 25.000 imagens. Para realizar a classificação, foram escolhidos modelos de aprendizado profundo de alto desempenho. O objetivo é avaliar a eficácia desses modelos no reconhecimento de padrões em imagens médicas. Espera-se que este estudo contribua para demonstrar o potencial da visão computacional no suporte ao diagnóstico médico, explorando a aplicação prática de redes neurais em cenários reais.

Revisão Bibliográfica

Segundo Carreras, Roncador, and Hamoudi 2024, a utilização de redes neurais convolucionais (CNNs) no diagnóstico de colite ulcerativa e câncer colorretal demonstra alta precisão, com acurácia acima de 99% na classificação dessas condições. A metodologia empregou aprendizado por transferência com a arquitetura ResNet-18, analisando marcadores imunológicos como LAIR1 e TOX2 para aprofundar a compreensão do microambiente imunológico dessas doenças.

Identify applicable funding agency here. If none, delete this.

Além disso, Sheriff et al. 2021 destacam que a arquitetura VGG-16 é eficaz na detecção precoce de câncer de pulmão, especialmente em combinação com técnicas de aumento de imagem, como rotação, espelhamento e ajuste de contraste. Essa abordagem melhora significativamente a precisão na classificação de imagens médicas e reduz a dependência de diagnósticos manuais, muitas vezes suscetíveis a erros.

Os dois estudos reforçam o papel do aprendizado profundo como ferramenta central no desenvolvimento de sistemas automatizados para diagnóstico médico, destacando a importância de arquiteturas robustas e pré-processamento eficaz de dados. Este trabalho baseia-se nessas abordagens para explorar o uso de CNNs em um contexto semelhante, buscando identificar padrões complexos em imagens médicas e melhorar a precisão diagnóstica.

Material e Métodos

Redes neurais convolucionais Usadas

Neste trabalho, foi utilizado três redes neurais para tarefas de classificação: **AlexNet**, **SqueezeNet** e **ResNet18**. Cada uma dessas arquiteturas tem características distintas que as tornam adequadas para diferentes cenários.

A **AlexNet** é composta por 8 camadas principais, sendo 5 camadas convolucionais seguidas por 3 camadas totalmente conectadas. AlexNet foi projetado para lidar com um grande número de parâmetros e utiliza técnicas como *ReLU* para funções de ativação e *dropout* para regularização.

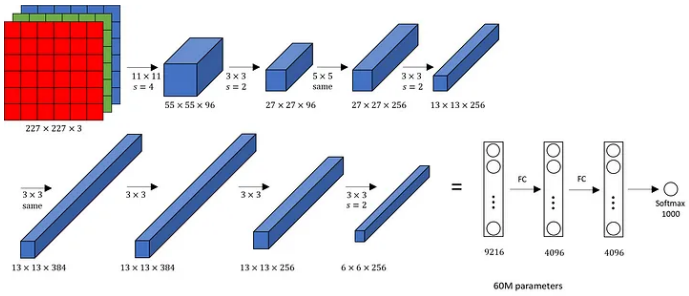

Fig. 1. Arquitetura AlexNet

- Camadas: 5 convolucionais + 3 totalmente conectadas

- Parâmetros: Cerca de 60 milhões
- Vantagens: Grande desempenho em tarefas de classificação de imagem.
- Desvantagens: Alto custo computacional e maior consumo de memória.

SqueezeNet é uma arquitetura de rede neural convolucional otimizada para minimizar o número de parâmetros, mantendo um bom desempenho. Ela introduz o conceito de **Fire Modules**, que combinam camadas de redução (*squeeze*) e expansão (*expand*) para reduzir custos computacionais sem comprometer a precisão.

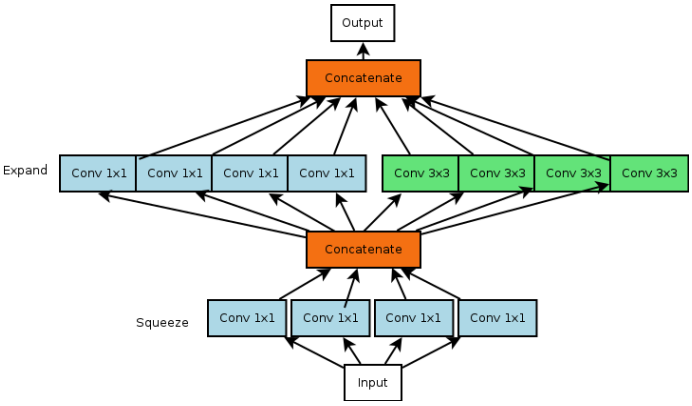

Fig. 2. Arquitetura SqueezeNet

- **Camadas**: Baseada em módulos Fire que otimizam a convolução reduzindo e expandindo canais.
- **Parâmetros**: Aproximadamente 1.2 milhões, significativamente menos que muitas arquiteturas populares.
- **Vantagens**: Pequeno tamanho do modelo, adequado para dispositivos com recursos limitados, e baixa latência para inferência.
- **Desvantagens**: Pode apresentar menor precisão em comparação com modelos mais complexos dependendo da aplicação.

A **ResNet18** é parte da família de Redes Residuais (ResNet), que introduz o conceito de *skip connections* ou conexões residuais. Estas conexões permitem que o gradiente flua de forma mais eficiente durante o treinamento de redes muito profundas, evitando o problema do desaparecimento do gradiente. A ResNet18, especificamente, é uma versão mais compacta da ResNet, composta por 18 camadas. Embora seja mais leve que as versões maiores da ResNet, ainda oferece excelentes resultados em termos de precisão.

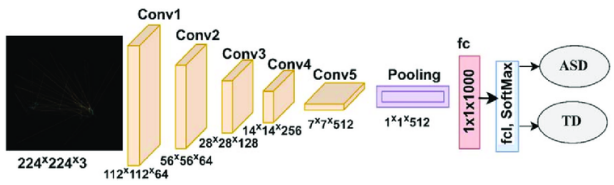

Fig. 3. Arquitetura ResNet18

- Camadas: 18 camadas com conexões residuais
- Parâmetros: Cerca de 11 milhões
- Vantagens: Bom desempenho em tarefas de classificação, capacidade de treinar redes profundas.
- Desvantagens: Requer mais poder computacional do que redes mais simples, como a EfficientNet-B1.

Conjunto de Dados

O conjunto de dados utilizado neste projeto é composto por 25.000 imagens, organizadas em cinco classes distintas, sendo 5.000 imagens em cada classe. Essas classes foram definidas de acordo com características específicas, conforme descrito a seguir:

- **Adenocarcinomas de cólon:** Tumores malignos localizados no cólon, caracterizados por células glandulares com crescimento desordenado, frequentemente invasivo.
- **Tecidos colônicos benignos:** Representações de tecidos saudáveis ou lesões benignas do cólon, sem características malignas, mantendo estruturas normais ou apenas levemente alteradas.
- **Adenocarcinomas de pulmão:** Tumores malignos nos pulmões, compostos por células glandulares, capazes de invadir tecidos pulmonares adjacentes.
- **Carcinomas de células escamosas de pulmão:** Um tipo de câncer pulmonar originado em células escamosas, frequentemente associado a fatores como tabagismo e exposição a substâncias carcinogênicas.
- **Tecidos pulmonares benignos:** Imagens de tecidos pulmonares saudáveis ou lesões benignas, incluindo áreas inflamatórias ou cicatrizes, sem evidências de malignidade.

Para organizar o conjunto de dados, a variável `diretorio_origem` foi configurada para armazenar o caminho das cinco pastas correspondentes às classes. A partir disso, foram aplicadas uma série de transformações e processos às imagens, visando adequá-las ao treinamento do modelo:

- **Redimensionamento:** As imagens foram redimensionadas para 224x224 pixels, tamanho padrão exigido por muitos modelos pré-treinados.
- **Conversão para tensor:** As imagens foram convertidas em tensores, estruturas de dados utilizadas pelo PyTorch.
- **Normalização:** As imagens foram normalizadas com médias e desvios padrão do conjunto de dados ImageNet ($[0.485, 0.456, 0.406]$ para médias e $[0.229, 0.224, 0.225]$ para desvios padrão), garantindo compatibilidade com os modelos pré-treinados.
- **Carregamento de dados:** A classe `datasets.ImageFolder` do PyTorch foi utilizada para organizar e carregar as imagens armazenadas em subpastas, onde cada subpasta representa uma classe.
- **Dataloader:** Um `DataLoader` foi configurado com tamanho de lote (*batch size*) de 64 e o parâmetro `shuffle=True`, garantindo que as imagens fossem embaralhadas a cada época para melhorar a generalização.

Divisão dos Dados

O conjunto de dados foi dividido em três subconjuntos:

- **Treinamento (70%):** Utilizado para ajustar os pesos do modelo.
- **Validação (15%):** Usado para monitorar o desempenho do modelo durante o treinamento e prevenir *overfitting*.
- **Teste (15%):** Utilizado para avaliar o desempenho final do modelo.

A divisão dos dados foi realizada de forma aleatória, com a semente fixa para garantir reprodutibilidade. A função `criar_dataloaders` realiza essa divisão e retorna três `DataLoaders`, um para cada subconjunto: treinamento, validação e teste. Cada `DataLoader` é responsável por carregar os dados em lotes (batches) durante o treinamento e avaliação do modelo.

Seleção e Inicialização do Modelo

A escolha do modelo a ser utilizado para o treinamento é feita por meio da função `menu_modelo`, que oferece ao usuário uma lista para inicializar os modelos.

Após a escolha , a função ajusta a camada final para o número de classes especificado no conjunto de dados.

A função `menu_modelo` realiza as seguintes etapas:

- Solicita ao usuário a escolha do modelo.
- Inicializa o modelo selecionado com pesos pré-treinados.
- Substitui a camada de saída para o número de classes do problema de classificação.
- Retorna o modelo pronto para ser treinado.

Se a opção selecionada for inválida, a função encerra o processo com uma mensagem de erro.

Hiperparâmetros

Os hiperparâmetros configurados no treinamento do modelo foram definidos como segue:

- **Tamanho do lote:** 64.
- **Taxa de aprendizado:** 0.001.
- **Momentum:** 0.9.
- **Número de épocas:** 50.

Configuração do Modelo

Para a seleção do modelo, foi implementado um menu interativo que permite ao usuário escolher entre diferentes arquiteturas de redes neurais convolucionais. Após a seleção, o modelo é transferido para o dispositivo disponível (`cuda:0` ou `cpu`), utilizando a função `torch.device`.

O modelo escolhido é configurado para treinar com o seguinte:

- **Função de perda:** `CrossEntropyLoss`, adequada para problemas de classificação.
- **Otimizador:** `SGD` (Gradiente Estocástico), configurado com os parâmetros de taxa de aprendizado e momentum.
- **Agendador de taxa de aprendizado:** Um `StepLR` é utilizado para ajustar a taxa de aprendizado em etapas fixas (`step_size=7` e `gamma=0.1`).

Treinamento, Validação e Teste do Modelo

A função `treinar_validar_e_testar` é responsável por realizar o treinamento, validação e teste do modelo. Ela recebe os `DataLoaders` para os conjuntos de treino, validação e teste, o modelo a ser treinado, o otimizador, a função de perda (loss function), o agendador de taxa de aprendizado (scheduler), o número de épocas de treinamento e o dispositivo (CPU ou GPU) onde o modelo será executado.

Durante o processo, as seguintes etapas são realizadas:

- **Treinamento:** Para cada época, o modelo é treinado utilizando o conjunto de dados de treino. A perda (loss) é calculada e o gradiente é retropropagado para atualizar os pesos do modelo. A acurácia do treino também é calculada.
- **Validação:** Após cada época de treino, o modelo é avaliado no conjunto de validação, calculando a perda e a acurácia. O agendador de taxa de aprendizado é então atualizado, se necessário.
- **Teste:** Ao final de cada época, o modelo também é avaliado no conjunto de teste para calcular a perda e a acurácia de teste, fornecendo uma medida do desempenho final do modelo.

Além disso, as métricas de perda e acurácia para treino, validação e teste são armazenadas para cada época, para monitoramento do desempenho do modelo ao longo do tempo.

O tempo total de treinamento e o tempo de cada época também são registrados e exibidos, fornecendo uma visão geral da eficiência do treinamento.

Ao final, a função retorna as listas de perdas e acurácias de treino, validação e teste para cada época.

Avaliação e Impressão dos Resultados

A função `avaliar_e_imprimir_resultados` é responsável por avaliar o desempenho do modelo no conjunto de validação. Ela calcula e imprime as métricas, como a matriz de confusão, o relatório de classificação e a acurácia.

Durante a avaliação, a função realiza as seguintes etapas:

- **Iteração sobre os lotes de validação:** Para cada lote no conjunto de validação, o modelo faz previsões e calcula as probabilidades para cada classe.
- **Cálculo de probabilidades:** As probabilidades de cada classe são calculadas utilizando a função softmax. Isso permite não apenas as previsões de classe, mas também a confiança do modelo em cada previsão.
- **Armazenamento dos resultados:** As classes reais e preditas, assim como as probabilidades, são armazenadas para posterior análise.
- **Métricas de Avaliação:** A função gera e imprime as seguintes métricas:
  - **Matriz de Confusão:** Mostra o desempenho do modelo ao classificar as amostras de cada classe. Cada linha representa uma classe real, e cada coluna, uma classe predita.
  - **Relatório de Classificação:** Fornece várias métricas de desempenho, como precisão (precision), recall, f1-score e suporte (support) para cada classe.

– **Acurácia:** A proporção de previsões corretas em relação ao total de previsões realizadas.

RESULTADOS

Relatório de Desempenho do Modelo AlexNet

Nos dados a baixo detalhamos as métricas calculadas para avaliar a eficácia do modelo AlexNet.

Matriz de Confusão (Validação)

A matriz de confusão para a validação é apresentada abaixo.

$$\begin{bmatrix} 741 & 0 & 0 & 1 & 0 \\ 0 & 718 & 2 & 0 & 0 \\ 0 & 4 & 743 & 0 & 0 \\ 0 & 0 & 0 & 786 & 0 \\ 0 & 0 & 0 & 0 & 755 \end{bmatrix}$$

Relatório de Classificação (Validação)

A seguir, apresentamos os valores para cada classe:

Classe	Precisão	Recall	F1-Score	Suporte
1	1.0000	0.9987	0.9993	742
2	0.9945	0.9972	0.9958	720
3	0.9973	0.9946	0.9960	747
4	0.9987	1.0000	0.9994	786
5	1.0000	1.0000	1.0000	755

Gráficos de Treinamento

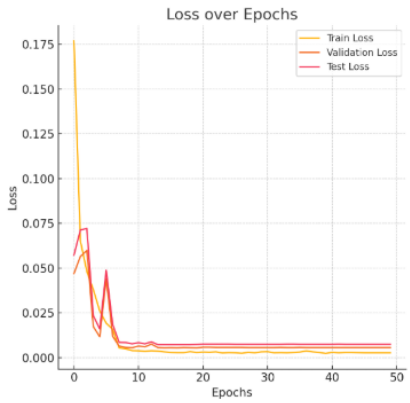

Fig. 4. Evolução da perda durante o treinamento do modelo AlexNet

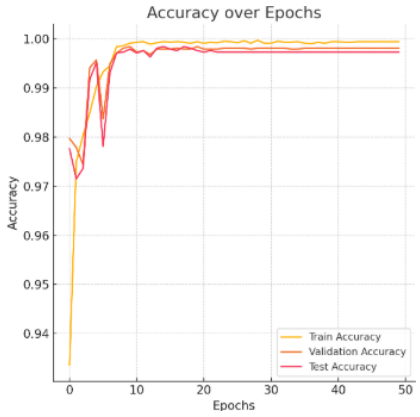

Fig. 5. Evolução da acurácia durante o treinamento do modelo AlexNet

Relatório de Desempenho do Modelo SqueezeNet

Nos dados abaixo detalhamos as métricas calculadas para avaliar a eficácia do modelo SqueezeNet.

Matriz de Confusão (Validação)

$$\begin{bmatrix} 742 & 0 & 0 & 0 & 0 \\ 0 & 717 & 3 & 0 & 0 \\ 0 & 2 & 745 & 0 & 0 \\ 0 & 0 & 0 & 786 & 0 \\ 0 & 0 & 0 & 0 & 755 \end{bmatrix}$$

Relatório de Classificação (Validação)

A seguir, apresentamos os valores para cada classe:

Classe	Precisão	Recall	F1-Score	Suporte
1	1.0000	1.0000	1.0000	742
2	0.9972	0.9958	0.9965	720
3	0.9960	0.9973	0.9967	747
4	1.0000	1.0000	1.0000	786
5	1.0000	1.0000	1.0000	755

Gráficos de Treinamento

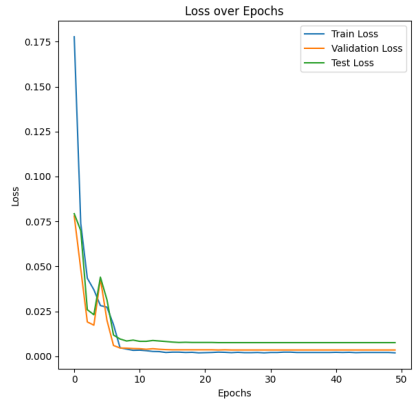

Fig. 6. Evolução da perda durante o treinamento do modelo SqueezeNet

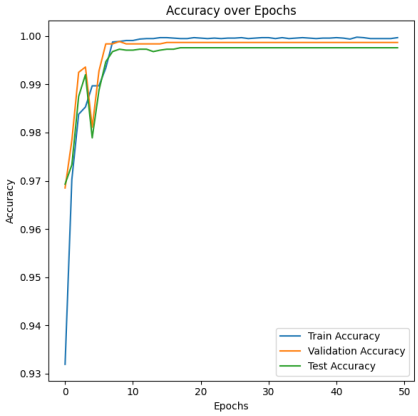

Fig. 7. Evolução da acurácia durante o treinamento do modelo SqueezeNet

A. Relatório de Desempenho do Modelo ResNet18

Nos dados abaixo detalhamos as métricas calculadas para avaliar a eficácia do modelo ResNet18.

Matriz de Confusão (Validação)

$$\begin{bmatrix} 741 & 0 & 0 & 1 & 0 \\ 0 & 718 & 2 & 0 & 0 \\ 0 & 4 & 743 & 0 & 0 \\ 0 & 0 & 0 & 786 & 0 \\ 0 & 0 & 0 & 0 & 755 \end{bmatrix}$$

Relatório de Classificação (Validação)

A seguir, apresentamos os valores para cada classe:

Classe	Precisão	Recall	F1-Score	Suporte
1	0.9887	1.0000	0.9993	742
2	0.9945	0.9986	0.9965	720
3	0.9987	0.9946	0.9966	747
4	1.0000	0.9987	0.9994	786
5	1.0000	1.0000	1.0000	755

Gráficos de Treinamento

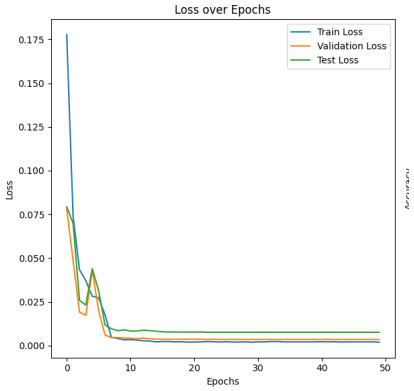

Fig. 8. Evolução da perda durante o treinamento do modelo ResNet18

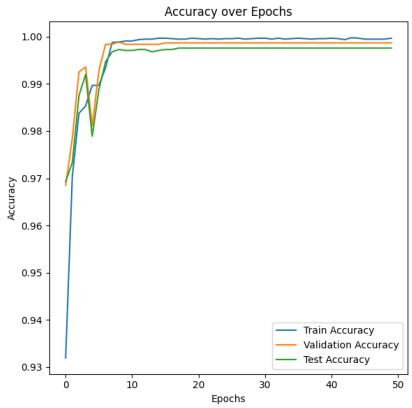

Fig. 9. Evolução da acurácia durante o treinamento do modelo ResNet18

Conclusão

Os experimentos realizados com os modelos AlexNet, SqueezeNet e ResNet18 evidenciaram a eficácia das redes neurais profundas na tarefa de classificação de imagens médicas. Cada modelo foi avaliado utilizando métricas como precisão, recall e F1-score, complementadas por gráficos que ilustram a evolução da perda e da acurácia durante o treinamento.

Entre os modelos avaliados, o SqueezeNet destacou-se como o mais eficiente, alcançando desempenho perfeito com valores de precisão, recall e F1-score iguais a 1.000 em todas as classes, apesar de sua arquitetura ser mais simples e compacta. Isso demonstra sua excelente capacidade de generalização e eficiência, mesmo com uma quantidade reduzida de parâmetros, o que o torna uma excelente opção para aplicações onde a leveza do modelo é uma prioridade.

O AlexNet também apresentou um desempenho notável, com métricas superiores a 0.99 em todas as classes. Sua arquitetura mais profunda e complexa permitiu um desempenho muito bom, com poucas falhas observadas na matriz de confusão. O AlexNet, embora mais pesado que o SqueezeNet, ainda se destacou pela sua eficácia em lidar com a variação e complexidade das imagens médicas.

Por outro lado, o modelo ResNet18, baseado na arquitetura residual, apresentou uma ligeira queda em precisão e recall para algumas classes específicas, especialmente em comparação com os outros dois modelos. No entanto, manteve-se competitivo, demonstrando sua robustez e eficiência, especialmente na capacidade de evitar problemas de degradação à medida que a profundidade da rede aumenta. O uso de conexões residuais, que permitem que as informações fluam mais facilmente por camadas mais profundas, mostrou-se valioso para preservar o desempenho nas camadas mais profundas, apesar das pequenas quedas observadas em certas classes.

Além dos resultados quantitativos, a análise da evolução da perda e da acurácia durante o treinamento revelou a estabilidade e a eficiência dos modelos em convergir para soluções ótimas. Os gráficos mostraram uma redução consistente na perda e um aumento significativo na acurácia, refletindo o aprendizado adequado das redes.

Os resultados quase perfeitos alcançados por todos os modelos podem ser atribuídos à qualidade do conjunto de dados utilizado, que foi bem curado e balanceado, e ao uso de pesos pré-treinados na ImageNet. Esse ponto de partida robusto permitiu que os modelos extraíssem características complexas e generalizassem bem, aproveitando o conhecimento prévio adquirido com a ImageNet.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.